DLR-IB-RM-OP-2020-97

Pose Estimation by Detection and Tracking of Artificial Markers for Planetary Exploration

Master's Thesis

Patrick Krömer

Deutsches Zentrum
DLR für Luft- und Raumfahrt

Institut für Robotik und Mechatronik	BJ.: 2020 IB.Nr.: DLR-IB-RM-OP-2020-97

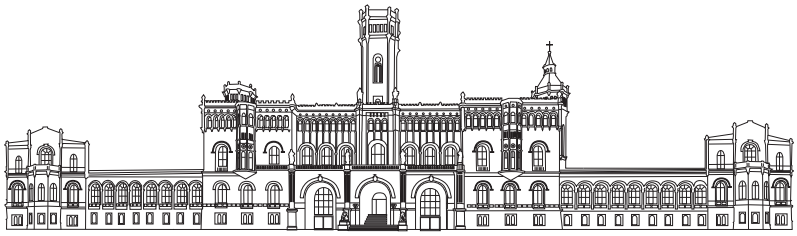
MASTERARBEIT

POSE ESTIMATION BY DETECTION AND TRACKING OF ARTIFICIAL MARKERS FOR PLANETARY EXPLORATION

Freigabe:	Der Bearbeiter:	Unterschriften
	Patrick Krömer	
	Betreuer:	
	Jongseok Lee	
	Der Institutsdirektor	alin.albu-schaeffer@dlr.de Digital signiert von alin.albu-schaeffer@dlr.de DN: CN=alin.albu-schaeffer@dlr.de Grund: Ich bin der Verfasser dieses Dokuments Ort: hier den Ort der Signierung ein Datum: 2020.10.29 21:02:55+01'00' Foxit PhantomPDF Version: 10.1.0
	Prof. Alin Albu-Schäffer	

Dieser Bericht enthält 91 Seiten, 46 Abbildungen und 3 Tabellen

Ort: Oberpfaffenhofen	Datum: 29-10-2020	Bearbeiter: Patrick Krömer	Zeichen:



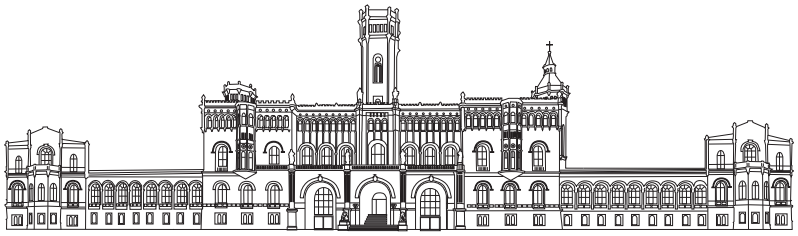
FACULTY OF CIVIL ENGINEERING AND GEODETIC SCIENCE

LEIBNIZ UNIVERSITY HANNOVER

Master's Thesis in Navigation and Field Robotics

Pose Estimation by Detection and Tracking of Artificial Markers for Planetary Exploration

Patrick Krömer



FACULTY OF CIVIL ENGINEERING AND GEODETIC SCIENCE

LEIBNIZ UNIVERSITY HANNOVER

Master's Thesis in Navigation and Field Robotics

Pose Estimation by Detection and Tracking of Artificial Markers for Planetary Exploration

Posenschätzung durch Detektion und Verfolgung von künstlichen Markern für die planetare Erkundung

Author:	Patrick Krömer
Supervisors (LUH):	apl. Prof. Dr. techn. Franz Rottensteiner M.Sc. Philipp Trusheim
Supervisors (DLR):	M.Sc. Jongseok Lee Dr. rer. nat. Wolfgang Stürzl
Submission Date:	30th June 2020

Institut für Photogrammetrie
und GeoInformation

Leibniz Universität Hannover

Institut für Photogrammetrie und GeoInformation,
Nienburger Straße 1, 30167 Hannover

Nienburger Straße 1, 30167 Hannover
Fakultät für Bauingenieurwesen und Geodäsie

Institut für Photogrammetrie
und GeoInformation

Prof. Dr.-habil. Christian Heipke

apl. Prof. Dr. techn. Franz Rottensteiner

Tel.+49 511 762-3893
Fax +49 511 762-2483
E-Mail: rottensteiner
@ipi.uni-hannover.de

28. October 2019

Topic of the M.Sc. Thesis of Mr Patrick Kroemer B.Sc.

Pose Estimation by Detecting and Tracking of Artificial Markers for Planetary Exploration (working title)

The cooperation of space rovers and drones will be of importance for future planetary exploration activities. Current state-of-the-art multi-copters are limited in their mission duration as they are typically limited in their payloads due to having to carry enough batteries. Therefore, one key technology is landing multi-copters on rovers. In this way, the mission duration of multi-copters can be increased, for instance by using a wireless charging pad on the rover. The precise and reliable landing of multi-copters on a mobile platform is a difficult task, because precise landing does not only depend on the accuracy of the landing spot estimation but also on the run-time of the involved algorithms. From a systems engineering perspective, the perception module must be fast enough to cope with the instability of aerial vehicles.

The goal of Mr. Krömer's thesis therefore is, to investigate improving pose estimation by combining different methods to an efficient pose estimation module. Basically, two methods will be used to accomplish this goal. The first is a detector for AprilTags, which detects squared markers with high gradients and reads their decoded message (payload number). This method gives a good pose estimation but at a low frequency. The other method used, is known as direct sparse odometry. It is an approach to estimate the trajectory, i.e. the sequence of 6D poses from the video stream of a moving camera. To find the corresponding pixels in consecutive frames, it uses the pixel intensities (direct; i.e by minimizing the photometric error) of a set of selected and widely distributed points (sparse). The direct sparse odometry needs depth information about the scene during initialization. This information can be provided by a depth image from stereo matching or, as envisaged in this project, by the AprilTag detector. The methods should be forged together in a manner that allows pose estimation with high accuracy and frequency so that it can be used for landing a small UAV (unmanned aerial vehicle) on a platform. An important constraint here is, that the computational power is also limited. Therefore, the algorithm should work efficiently and be restricted to the essentials.

The implementation of Mr. Krömer will be built on already existing pose estimation methods, but he will have to modify and combine them in a way that has to be investigated in the course of the project. The results should later be evaluated on a pre-recorded real-world dataset from a multicopter equipped with a stereo-camera. As ground truth, he will use a motion capturing system, that is based on 14 cameras and passive markers on the objects and the UAV.

apl. Prof. Dr. techn. Franz Rottensteiner / Philipp Trusheim, M.Sc.

Besucheradresse:
Nienburger Straße 1
30167 Hannover
www.ipi.uni-hannover.de

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources that were used to create this thesis are listed in the bibliography and acknowledged as references. This work was not previously presented to another examination board and has not been published yet.

Frankfurt am Main, 30^{th} June 2020

Patrick Krömer

Abstract

This thesis presents a new approach of indirectly tracking an static AprilTag with a monocular camera. The method developed, yields a metric pose estimation of the marker even when it leaves the cameras field of view. This was accomplished by using a keyframe-based direct odometry that was enriched by a spherical motion stereo algorithm in order to perform a 3D-to-2D tracking. The proper scaling of the transformation is achieved by a fusion of the depth maps from the marker and the motion stereo. Furthermore an error model was developed, which yields an uncertainty estimation for every pose, to allow the merging of pose estimations from other sources.
The work was done with consideration of an application to support a landing approach of a micro aerial vehicle in a real scenario. The results that were obtained during this work are promising to further develop this approach.

Keywords: Pose Estimation, Artificial Marker, Direct Sparse Odometry, Tracking, Spherical Motion Stereo

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The present thesis was carried out within the Perception and Cognition department of the Institute of Robotics and Mechatronics at the German Aerospace Center (DLR) in collaboration with the Institute of Photogrammetry and Geoinformation at the Leibniz University Hannover. At the Institute of Robotics and Mechatronics (RM) research is done on robots that can be used in places that are inaccessible or hard-to-reach for humans, like other planets or also terrestrial areas that were affected by natural disasters. This type of application sometimes requires a high degree of autonomy. While investigating how this can be made possible, the work at RM involves observing human behaviours in order to mimic them on a functional level to implement them in a robot. The Perception and Cognition department fulfils the task of environmental perception and the processing of the measured data to derive useful information for certain tasks; e.g. navigation or interaction with the environment.
The topic of this thesis is the estimation of the location and orientation of a camera to an stationary artificial marker by first detecting that marker and from then on tracking its relative pose to the camera. For the perception of this marker and the further tracking, a monocular camera is used. A possible scenario in which this topic is discussed is the deployment to an micro aerial vehicle (MAV). The MAV could be part of a robot team that is employed for planetary exploration. Since the robots in this team would be far away from human access, they have to perform some tasks autonomously, such as mapping the environment or, regarding a MAV, the landing on a platform. A current research project at the DLR, which demonstrates such a scenario and is a use case for this topic, is the *Helmholtz Future Project ARCHES* (e.g. Schuster et al. (2019); see also www.arches-projekt.de). Its focus is on creating a collaborative team of heterogeneous robots which are able to explore areas that are difficult to access for humans. Figure 1.1 shows a rover and the MAV *Ardea* that are part of ARCHES and were used for preliminary studies that were carried out at the beginning of the work. The thesis was done in close cooperation with the team that works on this project.

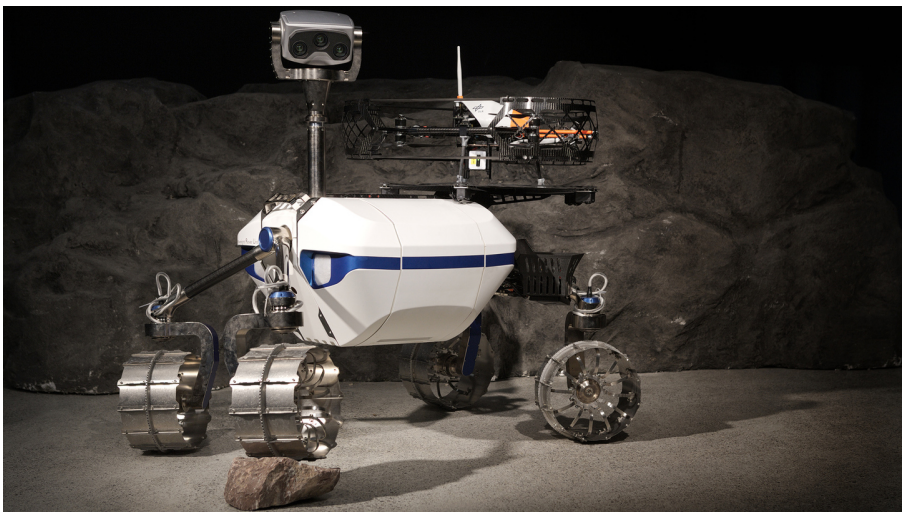


Figure 1.1: The multicopter Ardea standing on the landing-platform of the Lightweight Rover Unit (LRU) (Credits: DLR; CC-BY 3.0).

1.1 Perception and Cognition

For us humans, at and up-to a certain age, it is easy to navigate from one point to another relying on our sensors — like ears, eyes and the vestibular system — and the backend processing of the sensor output in our brain. To apply this biological trained methods to machines on the contrary, is a highly complex task and of paramount interest not just for research purposes in the robotic community. As the demand for mobile autonomous robots that act in a shared space with humans keeps growing, the sensors formerly used for sensing the robots state in a constrained area have to be able to perceive information from a highly dynamical environment. There are several types of exteroceptive sensors that can accomplish this task (and are also used in stationary robots), such as: cameras, laser scanners (lidar), sound navigation ranging (sonar) and radio detection and ranging (radar). This thesis only focuses on cameras, as the approach should be as general as possible which is facilitate by using a sensor that is cheap and widely used. To be more specific, a monocular camera will be used because it can also be employed on a very small robot that does not allow the installation of a stereo camera due to the fact that the baseline between the two cameras would be too small for useful stereo vision.
A lot of work was already done in the field of image processing which goes back to the early years of analogue photogrammetry. In 1840, before the term *photogrammetry* was coined, the french geodesist Arago used camera images to measure the shape of a landscape (Maybank, 1993). In the following years the process of land surveying was leveraged by photographs and maps were made with the help of aerial images. The fundamentals on which the photgrammetry is based on were already applied to perspective hand drawings and the roots of that principles even go further back in time, since they are basically

based on geometric assumptions.
A second branch of research which has a similar scope as photogrammetry, is computer vision. Although computer vision originated from research in biology and artificial intelligence, they have a lot of similar approaches nowadays. The main difference between them is only the underlying idea. Photogrammetry originated from the requirement to derive information from single images whereas most tasks in computer vision deal with consecutive image sequences as provided by video cameras.

1.2 Landing Spot Estimation

The cooperation of robots will be of importance for future planetary exploration activities. The *National Aeronautics and Space Administration (NASA)* for example, is already planning to launch the rotorcraft lander *Dragonfly* (Lorenz et al., 2018), as part of the mission with the same name, to explore Saturn's largest moon Titan. Dragonfly is a lander that is big enough to carry solar panels for recharging itself, whereas a micro aerial vehicle (MAV) like Ardea has to deal with the restrictions in terms of flight duration due to its low payload. It cannot carry enough batteries or solar panels for recharging to make a extensive exploration flight.
A solution to that problem, if the drone is part of a multi-robot team, is to land it on a wireless charging pad that is mounted on a rover (Brommer et al., 2018). A possible execution of this idea is shown in figure 1.2 and was given as a guiding principle that accompanies the entire thesis. This approach involves an accurate estimation of the position and orientation — in robotics often referred as *pose* — of that landing spot, which can be obtained by using artificial markers that are placed on the rover and which get captured in a camera image. However, approaching the pad in a stop and go manner by detecting the marker can be time consuming and is dependent on a continuous presence of the marker in the camera image if no further sensor or image based method is used to estimate the position and orientation of the MAV. Therefore the scope of the thesis is to elaborate a concept that fulfils the given requirements and which could be used to enrich a landing approach with the necessary information about the orientation and location of the landing platform.
The methodical approach that was chosen to follow this guideline is based on the decomposition of the overall problem. The central element of this approach has been identified as the estimation of the relative position and orientation from a camera to a point-of-interest (marked by an artificial marker). Furthermore, although the marker is theoretically movable, it is assumed to be static for the duration of the landing approach and therefore also for the concept described here. The initial estimation of that pose is obtained by a detector that identifies the marker in an image and leverages its prior

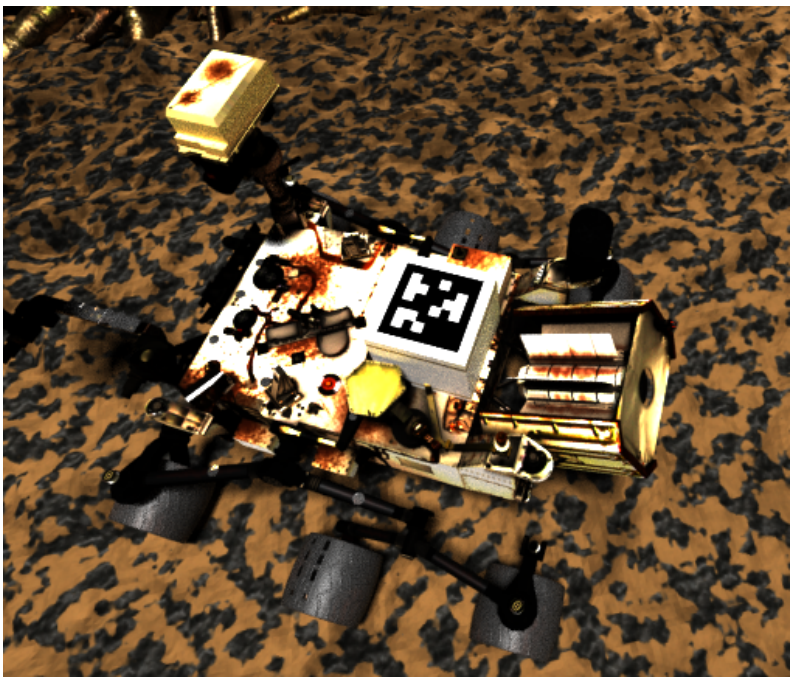


Figure 1.2: An image from the simulated dataset that was produced during the work on this thesis and shows the rover Curiosity with a landing platform mounted on top of it. The platform is marked by the black and white pattern, which is called an AprilTag.

knowledge of the parameters of this marker to calculate the pose.
The last and major part of the task, the tracking, is included by an indirect tracking approach. Since the marker is not moving, using one of the classic filter methods is not necessarily required to track it. This makes it possible to use the additional information that is located in the image areas around the marker instead of just focusing on the small area of the tag. For this purpose a keyframe-based visual odometry that uses 3D-to-2D point correspondences is included to the work. A depth map that is obtained by the fusion of a depth map calculated by using the tag position and one that is calculated using a motion stereo approach, yields the depth information, which the visual odometry needs for creating keyframes.
These individual parts are then combined into an overall concept, in which a monocular visual odometry was used as a framework that was enriched by a motion stereo algorithm and an interface for receiving external information from the marker detector. By combining the pose estimation coming from the marker detector with the relative pose coming from the modified visual odometry, a combined estimation of the markers current position and location can be made.
It is therefore possible to indirectly estimate the markers pose even when it cannot be directly measured by the detector, provided that an initial detection is available. The

methodologically interesting option that arises from this concept is that not only the tag is tracked, but also a monocular visual odometry with properly scaled pose estimations is developed.

1.3 Related Work

While researching on how to establish a working approach for the given task, several different topics from robotics, photogrammetry and computer vision were taken into consideration as solution. Work already done by others and related to that topics are referred and briefly explained below. In addition, it is explained which areas are of special interest for this thesis and therefore will be deepened in the following chapters.

1.3.1 Artificial Marker

An artificial (or *fiducial*) marker is an object of known size and shape that is place into a scene as a benchmark. When taking an image of the scene with the artificial object inside, the absolute location and orientation of the object to the camera or vice versa can be estimated (Hartley and Zisserman, 2003). This principle is used by the *AprilTag* detector of Olson (2011), of which meanwhile a revised third version exists (Krogius, Haggenmiller, and Olson, 2019). The drawback of this type of detectors is that they become slow when applied to images with a high resolution as pointed out in the work of Lee et al. (2020). Lee proposed a marker tracking algorithm with a visual-inertial odometry that estimates the pose of the tag, if no markers are detected. The approach uses multiple markers for resolving ambiguities and works with the *ARToolKitPlus* detector (Wagner and Schmalstieg, 2007). AprilTags are also used to yield a ground truth for visual datasets that were recorded in large outdoor areas, where it would be difficult to establish a traditional method which is based on camera arrays (Pfrommer et al., 2017). There are also several other algorithms that make use of artificial markers, and differ in speed, robustness and accuracy. In this work, the decision was made in favour of the AprilTag mainly because the pose estimations provided by its algorithm are among the most accurate (Abbas et al., 2019) — an example of it is shown on top of the rover in figure 1.2.

1.3.2 Pose estimation

For continuously estimating the ego-motion of a camera (pose), there are many approaches and developments. Some of them reach far back in time, but still today, it is a research area of high interest as the amount of recent works shows.
The principle of continuous pose estimation that is used in this work belongs to the

branch of monocular visual odometries. A further subdivision inside that area can be done by separating them into *feature-based* and *appearance-based* methods which differ in the way they extract information from the images. Each of them also can use a different principle of tracking those points over time. Either it is by tracking the points in consecutive images (Howard, 2008) or by tracking the points relative to a keyframe (Engel, Koltun, and Cremers, 2018a). The latter belongs to the appearance-based methods whereas the former is a feature-based visual odometry. A recent development of the direct sparse odometry (DSO) of Engel was done by Fontan, Civera, and Triebel (2020), who extended the approach by a information theory driven selection of the points to reduce their amount. This has the benefit that the runtime of the algorithm is reduced while also eliminating outliers.
If the focus of estimating the ego-motion is extended to simultaneously creating a map of the environment, the method is called simultaneous localisation and mapping (SLAM). VO and SLAM both belong to the much older structure from motion (SfM) concept, which targets the problem of simultaneously estimating the motion of the camera while using that motion to estimate three-dimensional object points (structure) from the two-dimensional image points. While SLAM is used synonymous to SfM, visual odometry is a particular case of SfM, where the pose estimation is emphasised (Scaramuzza and Fraundorfer, 2011).
AprilTags were already used in simultaneous localisation and mapping (SLAM) approaches for loop closing together with a visual odometry and a pose graph optimisation by Pfrommer and Daniilidis (2019) and Munoz-Salinas, Marin-Jimenez, and Medina-Carnicer (2019). To work properly, the SPM-SLAM by Munoz-Salinas, Marin-Jimenez, and Medina-Carnicer (2019) needs at least two tag detections in one image and the TAG-SLAM by Pfrommer and Daniilidis (2019) needs previously known locations of the tags.

1.4 Scope of the thesis

The scope of this thesis is to present a proof-of-concept for the tracking of an static artificial marker. This was accomplished by an indirect tracking of the marker by modifying the *Information-Driven direct RGB-D Odometry* of Fontan, Civera, and Triebel (2020) as described in chapter 3. This visual odometry was chosen after researching about the different types of VO and a careful consideration of the respective advantages and disadvantages. The final decision to choose this odometry depended on the advantages of fast tracking by reducing the number of points without great loss of accuracy and the planned future use of this VO in the multi-copter Ardea. The approach was then tested on the simulated dataset described in section 3.3, which also was created during the work

on this thesis. The results of these tests are presented and evaluated in chapter 4.
To sum up, in this work a method is developed which fulfils the requirement to track an artificial marker and also provides an extension for an existing visual odometry. The whole concept uses only a monocular camera and a physical artificial marker. Furthermore, a model for estimating the error is developed which yields a uncertainty for each pose estimation and makes it possible to evaluate the estimations. This uncertainty can be used when merging pose estimations from different sources or, within the fictitious setup of a landing approach, to estimate the probability of a successful landing.
This thesis extends the aforementioned works by a setup that requires only one sensor and one AprilTag for the pose estimation. Due to the implemented motion stereo it is also suitable for aerial vehicles that operate at further away distances from the ground, where stereo cameras normally cannot yield a good depth estimation due to the relation between the baseline of the cameras and the distance to the object.

Chapter 2

Fundamentals

In this chapter the fundamentals necessary to follow the statements in this work are provided. The first section deals with the mathematical and geometrical fundamentals necessary for understanding and describing the motion of an object trough the three-dimensional space. The following sections focus on methods of image processing that were used in the implementation of the approach.
In section 2.4, which elaborates the operation principles and applications of visual odometry, a more detailed insight into the two most commonly used methods is given. Although one method was ultimately preferred to the other, it was an essential part of the work process to understand both methods fundamentally in order to then choose the more appropriate one. Section 2.5 introduces AprilTag, an artificial marker that is widely used in robotics and is followed by the explanation of motion stereo in section 2.3. Finally a brief presentation of an unmanned aerial vehicle that was used as template for the experimental realisation is given in section 2.6.

2.1 Transformations

The arbitrary motion of an object in the three-dimensional space $\mathbb{R}^3$ can be defined as a function f that maps the object from $\mathbb{R}^3$ to $\mathbb{R}^3$ as represented in equation (2.1) (Blanco Claraco, 2010).

$$f : \mathbb{R}^3 \to \mathbb{R}^3 \tag{2.1}$$

$\mathbb{R}^3$ is the Cartesian product of the real n-space $\mathbb{R}^n$ for $n = 3$ which is also denoted as the *Euclidean space* (eqn. (2.2)).

$$\mathbb{R}^3 = \mathbb{R} \times \mathbb{R} \times \mathbb{R} \tag{2.2}$$

The objects where the focus is on are rigid, thus the function f must preserve the structure of an object when it is mapped. Therefore the function f is a *homomorphism* and because the rigid body motion can be reversed, more precisely an *isomorphism*. The motion of an object in 3D-space has 6 degrees of freedom (DoF) and consists of a rotation and a

translation, together denoted as *Euclidean transformation* (Hartley and Zisserman, 2003). Let us first focus on the rotations.

As Blanco Claraco (2010) notes, three-dimensional rotations can be expressed as a set of 3×3 matrices that belong to the special orthogonal group $\mathbf{SO}(3)$ which is a subset of the general linear group $\mathbf{GL}(3, \mathbb{R})$ (eqn. (2.3a)). The members of this group fulfil all the conditions mentioned in equation (2.3). The vector elements of the matrix are orthogonal to each other, which means that a square matrix multiplied by its transpose yields the identity matrix $\mathbf{I}_n$ (eqn. (2.3b)) and that its transpose is equal to its inverse (eqn. (2.3c)). Furthermore the determinate has to be 1 (eqn. (2.3d)), as this is the property of a orthogonal matrix that is a pure rotation without reflection, also known as *proper isometry*.

$$\mathbf{R} \in \mathbf{SO}(3) \subset \mathbf{GL}(3, \mathbb{R}) \tag{2.3a}$$

$$\mathbf{R} \cdot \mathbf{R}^{\mathsf{T}} = \mathbf{R}^{\mathsf{T}} \cdot \mathbf{R} = \mathbf{I}_3 \tag{2.3b}$$

$$\mathbf{R}^{-1} = \mathbf{R}^{\mathsf{T}} \tag{2.3c}$$

$$\det(\mathbf{R}) = 1 \tag{2.3d}$$

As an example a point $\mathbf{x_1} = (x_1 \quad y_1 \quad z_1)^{\mathsf{T}}$ that is mapped to a point $\mathbf{x_2} = (x_2 \quad y_2 \quad z_2)^{\mathsf{T}}$ is considered. This is done by applying the mapping function $f = \mathbf{R}_{21}$ to point $\mathbf{x_1}$ as shown in equation (2.5), following the linear maplet in equation (2.4).

$$\mathbf{x} \mapsto \mathbf{R} \cdot \mathbf{x}, \qquad \mathbf{x} \in \mathbb{R}^3, \quad \mathbf{R} \in \mathbf{SO}(3) \tag{2.4}$$

$$\mathbf{x_2} = \mathbf{R}_{21} \cdot \mathbf{x_1} \tag{2.5}$$

To map the resulting point $\mathbf{x_2}$ back to $\mathbf{x_1}$, it can be multiplied to the inverse respectively transpose of $\mathbf{R}_{21}$:

$$\mathbf{x_1} \;=\; \mathbf{R}_{21}^{-1} \cdot \mathbf{x_2} \;=\; \mathbf{R}_{21}^{\mathsf{T}} \cdot \mathbf{x_2} \;=\; \mathbf{R}_{12} \cdot \mathbf{x_2} \tag{2.6}$$

Including translations to this expression in order to describe a 6 DoF transformation $\mathbf{T}$ is not straightforward, since three-dimensional translations are non-linear transformations in $\mathbb{R}^3$. They follow the maplet in equation (2.7) which shows that the application of a translation to a given vector $\mathbf{x}$ is the addition of the transitional vector $\mathbf{v}_0$ to that given vector $\mathbf{x}$.

$$\mathbf{x} \mapsto \mathbf{x} + \mathbf{v}_0, \qquad \mathbf{x}, \mathbf{v}_0 \in \mathbb{R}^3 \tag{2.7}$$

In order to include translations to the linear expression, the expression is extended by one dimension to the so called *homogeneous coordinates* (Möbius, 1827). That means that the three-dimensional point $\mathbf{x}_1$ is extended by a homogeneous coordinate. In most applications this always will be the unity (eqn. (2.8a)). The advantage of this extension is

that the translation becomes a linear transformation and the whole 6 DoF transformation $\mathbf{T}$ can be expressed as shown in equation (2.8b).

$$\begin{pmatrix} \mathbf{x}_2 \\ 1 \end{pmatrix} = \mathbf{T} \begin{pmatrix} \mathbf{x}_1 \\ 1 \end{pmatrix} \tag{2.8a}$$

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix} = \left(\begin{array}{ccc|c} & \mathbf{R} & & \begin{matrix} t_x \\ t_y \\ t_z \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix} \tag{2.8b}$$

This is equal to a matrix multiplication with $\mathbf{R}$ followed by a vector addition of $\mathbf{t}$:

$$\mathbf{x}_2 = \mathbf{R}\mathbf{x}_1 + \begin{pmatrix} t_x & t_y & t_z \end{pmatrix}^\mathsf{T} \tag{2.9}$$

Because the transformation now can be expressed as a matrix multiplication which is subject to the associative property, multiple transformations can easily be connected instead of executing them one after the other (eqn. (2.10)).

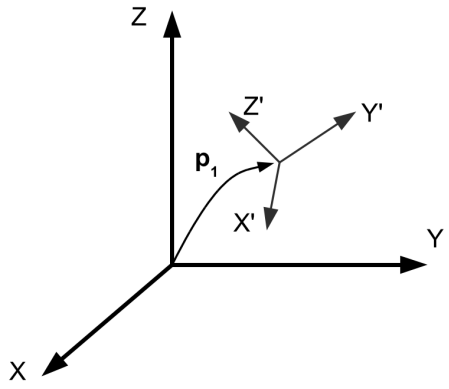
$$f(g(h(\mathbf{x}))) = (f \circ g \circ h)(\mathbf{x}) \tag{2.10a}$$

$$f(\mathbf{x}) = \mathbf{A}\mathbf{x}, \quad g(\mathbf{x}) = \mathbf{B}\mathbf{x}, \quad h(\mathbf{x}) = \mathbf{C}\mathbf{x} \tag{2.10b}$$

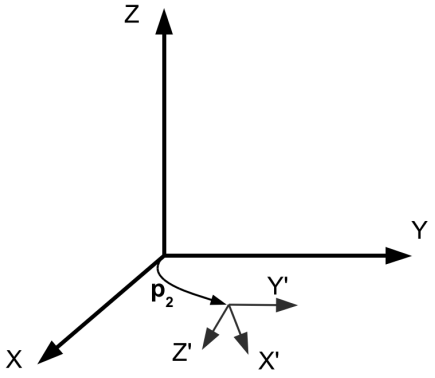
$$\mathbf{A}(\mathbf{B}(\mathbf{C}\mathbf{x})) = (\mathbf{A}\mathbf{B}\mathbf{C})\mathbf{x} \tag{2.10c}$$

In equation (2.8b) a invertible 4×4 matrix was presented, which by definition belongs to $\mathbf{GL}(4, \mathbb{R})$ and not $\mathbf{GL}(3, \mathbb{R})$, but in the particular case of the so defined set of transformation matrices T (along with the group operation of matrix product), they form the group of affine rigid motions which, with *proper* rotations, is denoted as the special Euclidean group $\mathbf{SE}(3)$ (Blanco Claraco, 2010). $\mathbf{SE}(3)$ is, as well as $\mathbf{SO}(3)$, a Lie group and this group is often referred to as *poses* with 6 DoF. Figure 2.1 shows an example of how two adjacent poses can be combined to one pose with respect to the world frame (X, Y, Z). The transformations that are consider here always consist of a rotation and a translation. For the translation, there is always an unambiguous expression in the form of a 3-dimensional vector, where the rotation can be described in different ways. The four most common ones are:

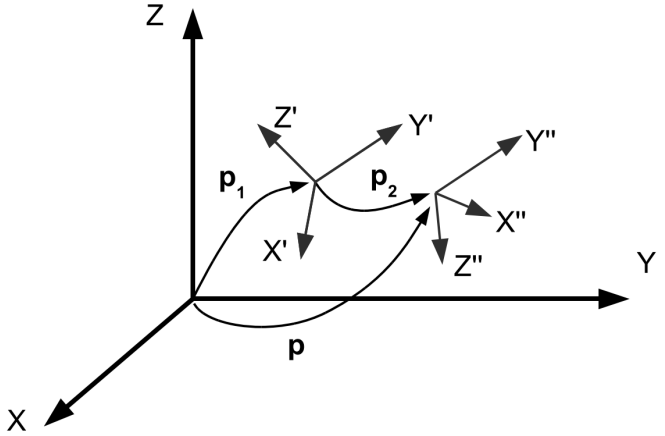
1. The just described **rotational matrices**,
2. **Euler angles**,
3. **quaternions** and
4. the **axis–angle representation**.



(a) The pose $\mathbf{p}_1$



(b) The pose $\mathbf{p}_2$



(c) The pose $\mathbf{p}$ as composition of pose $\mathbf{p}_1$ and $\mathbf{p}_2$

Figure 2.1: The composition of the two 6D poses $\mathbf{p}_1$ and $\mathbf{p}_2$ leads to $\mathbf{p}$ (reprinted and modified from Blanco Claraco, 2010).

For presenting results, the Euler angles are well suited, because it is easier to understand their representation, whereas rotational matrices, although including redundant information are well suited for combining different transformations. The equivalences between the different representations, their transformations among each other and their uncertainties and drawbacks are extensively elaborated in the work of Blanco Claraco (2010).

2.1.1 Errors in transformation

There are several possibilities to express the error of a transformation, for example by splitting them up in two separate transformation or by directly adding the error of every parameter to it. The representation of errors in translation and rotation used in this thesis is defined as shown in equation (2.11).

$$\mathbf{T} = \begin{pmatrix} \Delta\mathbf{R}\mathbf{R}_0 & \mathbf{t}_0 + \Delta\mathbf{t} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} (\mathbf{1} + [\Delta\mathbf{r}]_\times)\mathbf{R}_0 & \mathbf{t}_0 + \Delta\mathbf{t} \\ 0 & 1 \end{pmatrix} \tag{2.11a}$$

$$= \begin{pmatrix} \mathbf{R}_0 + [\Delta\mathbf{r}]_\times\mathbf{R}_0 & \mathbf{t}_0 + \Delta\mathbf{t} \\ 0 & 1 \end{pmatrix} \tag{2.11b}$$

$$= \begin{pmatrix} \mathbf{R}_0 & \mathbf{t}_0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} [\Delta\mathbf{r}]_\times\mathbf{R}_0 & \Delta\mathbf{t} \\ 0 & 1 \end{pmatrix} \tag{2.11c}$$

$\mathbf{T}$ is the erroneous transformation that was measured or estimated, $\Delta\mathbf{R}$ and $\Delta\mathbf{t}$ the errors in rotation respectively translation, $\mathbf{R}_0$ and $\mathbf{t}_0$ the true rotation respectively translation and $[\Delta\mathbf{r}]_\times$ the cross-product operator for small rotations that builds a 3×3 skew-symmetric matrix from a 3×3 vector like shown in equation (2.12a).

$$\mathbf{R} \simeq \begin{pmatrix} 1 & -\Delta\Phi & \Delta\Theta \\ \Delta\Phi & 1 & -\Delta\Psi \\ -\Delta\Theta & \Delta\Psi & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & -\Delta\Phi & \Delta\Theta \\ \Delta\Phi & 0 & -\Delta\Psi \\ -\Delta\Theta & \Delta\Psi & 0 \end{pmatrix}}_{[\Delta\mathbf{r}]_\times} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.12a}$$

2.2 Image Processing

2.2.1 Pinhole Camera Model: Simplify the Capturing of Light

The pinhole camera model is a simplified model of the working principle of a camera. It describes the relationship between three-dimensional object points and its projection on the two-dimensional image plane. In this model it is assumed that the centre of the camera is a pinhole and all light-rays that are captured on the image plane have to pass that point. It is a description of the natural phenomenon that occurs when light enters a

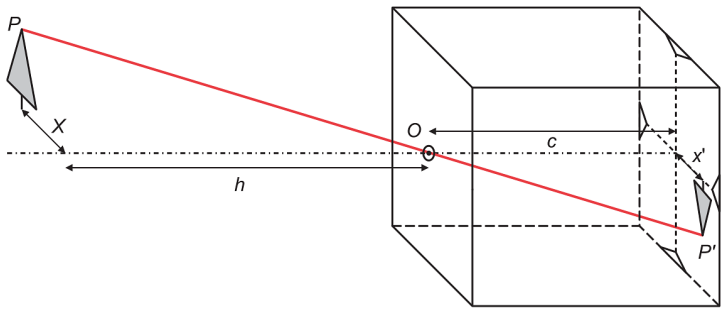


Figure 2.2: The pinhole camera (reprinted from Gillies, 2015).

dark room that just has a small point shaped opening and a image of the outside scene is projected upside-down at one of the walls as shown in figure 2.2. This is also known as *camera obscura*.

In a real perspective camera that opening is a lens that is used to amplify the intensity of image points by focusing light that emerges in different directions from one object point to its corresponding image point on the image plane. As already mentioned, the pinhole camera model is just a simplification because it does not take into account that a lens can cause distortions and a camera has a limited aperture. However this is a widely used model that can be used in most cases as approximation of the real process.

The geometrical relations between the object point $\mathbf{X}$ and its corresponding image point $\mathbf{x}$ are shown in figure 2.3. The relation between those two points is expressed by equation (2.13), where f is the focal length of the camera and $(X, Y, Z)^\mathsf{T}$ are the three coordinates of the object point in the camera frame.

$$(X, Y, Z)^\mathsf{T} \mapsto (f\frac{X}{Z}, f\frac{Y}{Z}, f)^\mathsf{T} \tag{2.13}$$

Figure 2.3 also shows that the image plane is defined to be in front of the camera centre in the pinhole model. This convention is made, because it is more convenient to have the same signs on both sides of the central projection mapping function (eqn. (2.13)). The relation of the object point to the image point can also be expressed as a matrix multiplication when using homogeneous coordinates (eqn. (2.14)).

$$\begin{pmatrix} f_x X \\ f_y Y \\ Z \end{pmatrix} = \begin{pmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.14}$$

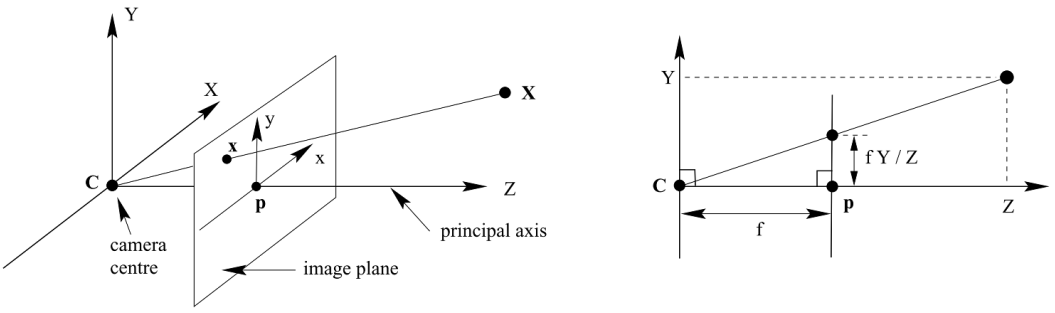


Figure 2.3: The pinhole camera model (reprinted from Hartley and Zisserman, 2003).

In most cases the focal lengths in x-direction f_x and in y-direction f_y are equal, because the pixel on the image plane are quadratic. To express the image points in the image frame, the expression in equation (2.14) has to be extended by a translation part $(p_x, p_y)^\mathsf{T}$, because the origin of the image coordinate frame is defined as the lower right corner (when looking from the camera centre to the image plane) of the image plane in this example. This yields equation (2.15) with the camera calibration matrix $\mathbf{K}$ (eqn. (2.16)) as part of the projection matrix. The calibration matrix is shown with all five intrinsic parameters. In most applications the skew symmetric parameter s which denotes the skew coefficient between the x- and y-axis is zero. The parameters in the matrix $\mathbf{K}$ are determined by performing a camera calibration.

$$\begin{pmatrix} f_x X + Z p_x \\ f_y Y + Z p_y \\ Z \end{pmatrix} = \left(\mathbf{K} \,\middle|\, \mathbf{0} \right) \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.15}$$

where

$$\mathbf{K} = \begin{pmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.16}$$

If the inverse of the calibration matrix $\mathbf{K}$ is applied to the image point $\mathbf{x}$ it yields the point $\hat{\mathbf{x}}$ which is the image point $\mathbf{x}$ expressed in normalised coordinates.

2.3 Spherical Motion Stereo

Motion Stereo is a case of stereo matching, where the motion that is expressed by a transformation $\mathbf{T}$ between two camera positions is used to calculate the relative distances of object points to the camera centre. The essential matrix $\mathbf{E}$ which describes the relation

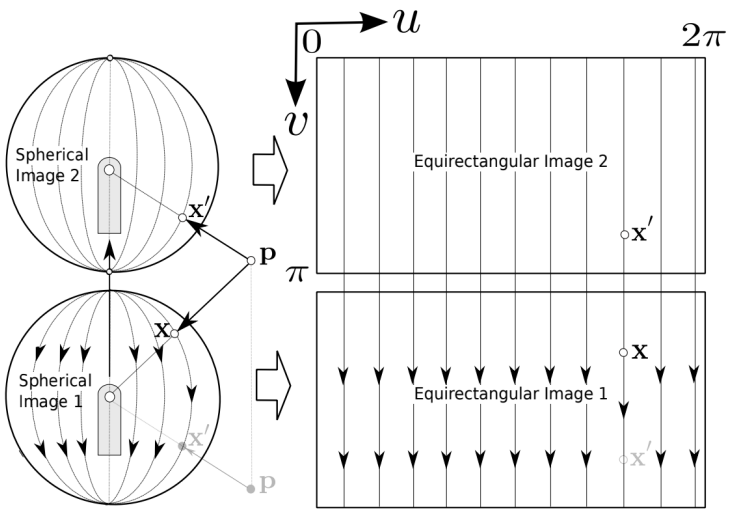


Figure 2.4: Mapping a spherical image to equirectangular images results in vertical parallel epipolar lines (reprinted from Pathak et al., 2016).

between two corresponding image points, can be calculated like shown in equation (2.17).

$$\mathbf{T} = \left(\begin{array}{ccc|c} & \mathbf{R} & & \mathbf{t} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{2.17a}$$

$$\mathbf{E} = \mathbf{R}[\mathbf{t}]_\times \tag{2.17b}$$

$$[\mathbf{t}]_\times = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} \tag{2.17c}$$

Together with the known camera calibration matrix $\mathbf{K}$ and the epipolar constraint shown in equation (2.18) can then be solved to find corresponding points in the images.

$$\mathbf{x_2}^\mathsf{T}\mathbf{K}^{-\mathsf{T}}\mathbf{E}\mathbf{K}^{-1}\mathbf{x_1} = 0 \tag{2.18}$$

where $\mathbf{x_1}$ and $\mathbf{x_2}$ are corresponding points in the two images and $\mathbf{K}^{-\mathsf{T}}\mathbf{E}\mathbf{K}^{-1}\mathbf{x_1}$ describes the epipolar line in image 2 on which the corresponding point x_2 can be found. A pair of corresponding image points are two points in two different images that are the projection of the same object point to the respectively image plane. The point at which all epipolar lines intersect is the epipole, which also corresponds to the translation direction (Hartley and Zisserman, 2003). If the motion is a pure translation perpendicular to the camera z-axis, the epipole is at infinity and all epipolar lines are therefore aligned parallel to each other. This makes the search for corresponding points much easier, because the

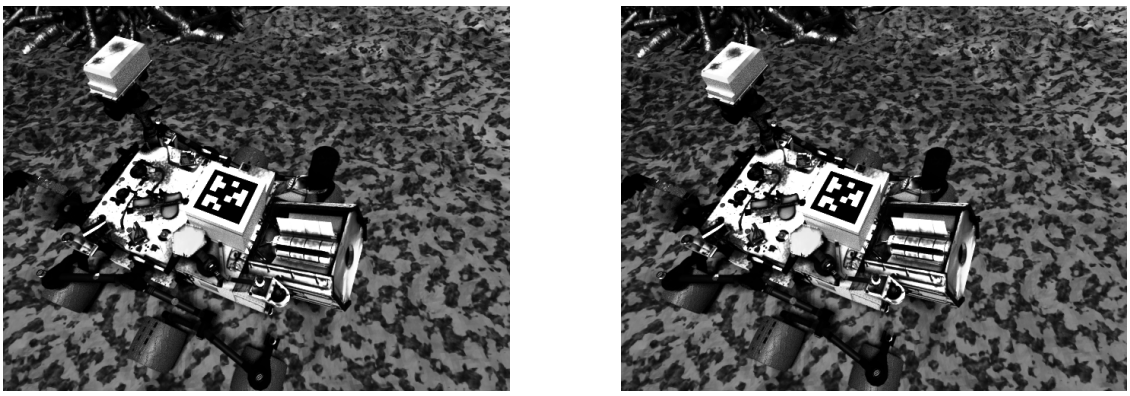


Figure 2.5: The input images for the spherical motion stereo algorithm.

epipolar lines do not have to be calculated for every point individually. Furthermore it is possible to use the horizontal shift of corresponding points to determine the distance of the corresponding object point to the camera. This horizontal shift is called *disparity* in computer vision and is also known as horizontal parallax (Kraus, 2007). The disparity is inversely proportional to the depth.
From the disparity $d = u_l - u_r$ of the horizontal location of the point $\mathbf{p_l}$ in the left respectively the point $\mathbf{p_r}$ in right image, the distance Z of the object point to the image plane can be calculated with equation (2.19) where B is the baseline between the two images.

$$Z = \frac{f \cdot B}{d} \tag{2.19}$$

If the Transformation between the two images is not known by scale, also the distance Z will only be estimated up-to scale.
It is also possible to determine the shift of a pixel in two tilted images by taking the horizontal and vertical parallax into account. This can be done, if only the depth information of a few points are of interest, whereas it requires much more computational effort if a dense depth map should get estimated. It is therefore the usual way to rectify the images before starting the stereo matching. However, if the motion is mainly in z-direction of the camera, the epipoles in both images are at almost the same position and rectifying the image using cartesian coordinates will fail due to shifting the epipole to infinity also would make the images infinitely large. Abraham and Förstner (2005) suggested a method that is similar to Pollefeys, Koch, and Van Gool (1999) uses spherical coordinates and works for movements in all directions. This method works by first mapping each of the two images to a sphere so that the two resulting epipolar-axis are aligned. The spherical images are then mapped to an equirectangular image as shown in figure 2.4. Corresponding points are then vertically aligned. The disparity and depth map are determined in this representation before the image is mapped back to cartesian coordinates. An example of this process is shown in figure 2.6 for the input images in figure 2.5.

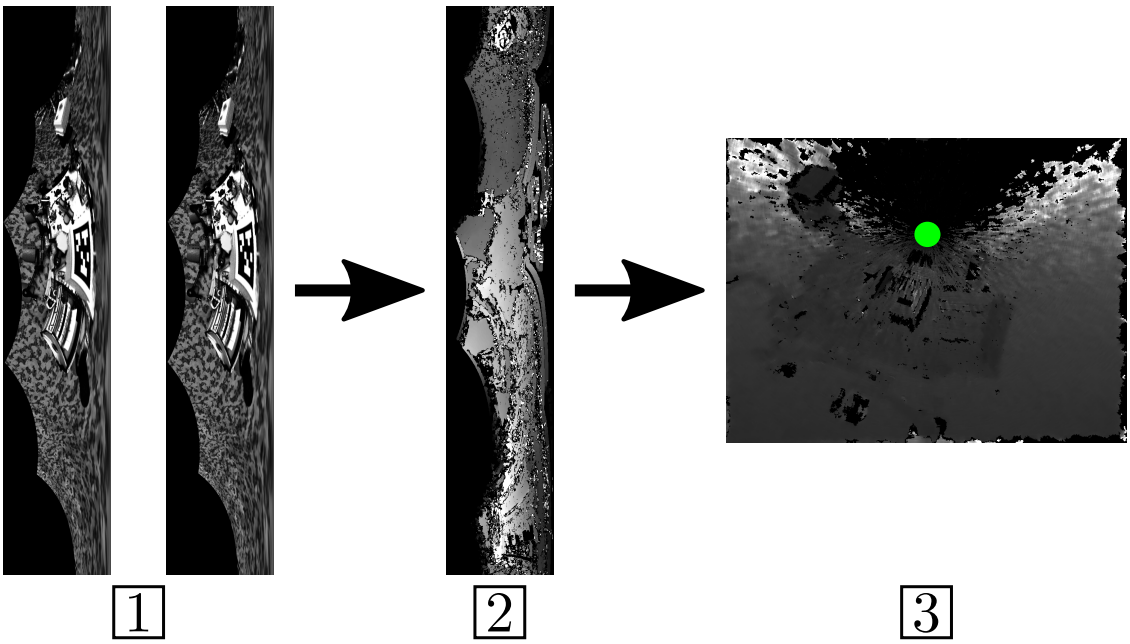


Figure 2.6: The equirectangular mapping of the two images (1), the calculated disparity map (2) and the resulting depth map in cartesian coordinates (3). Note the noisy depth values close to the epipole (green) above the rover. The equirectangular images were rotated by 90° for presentation purposes.

2.4 Visual Odometry

Odometry is a technique to incrementally measure the travelled distance and direction of a moving object with an internal sensor. The word originates from the two greek words *hodós*, which means path and *métron*, which means measurement. In a car for instance, the odometer keeps track of how many kilometres the car travelled altogether, but it will not show the current position and orientation to a reference frame. To allow for the estimation of the current position and orientation - in robotics and computer vision also often just called *pose* - odometry makes use of more than just simply summing up all distances. Odometry for a car could make additional use of the steering angle or the difference in covered distance of the right and left wheel. In shipping navigation a similar concept is applied with the help of a compass and the measurement of velocity of the ship. The velocity integrated over time yields to the travelled distance. Both aforementioned methods are a type of dead reckoning where the full path of the agent is getting integrated over time.

The expansion of this term to visually measurable movements was first mentioned by Srinivasan et al. (1996). Srinivasan discovered that ”bees possess a visually driven ‘odometer’”, which makes use of the optical flow they perceive when flying. However, the concept is much older than the term. One of the first human-made applications of this method was carried out by Moravec (1981) for the Stanford AI Lab cart (figure 2.7).

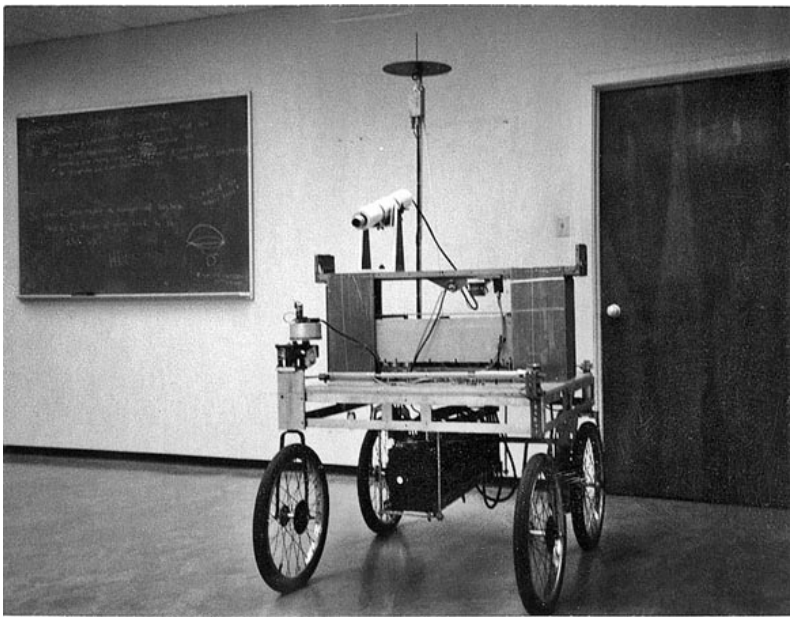


Figure 2.7: The Stanford AI Lab cart with the camera (white tube) on the slider on top of the cart (reprinted from Moravec, 1980).

He used digital cameras to find corresponding pixels in consecutive images by matching them. The *slider stereo* system he designed for this purpose took several overlapping images at every position and looked for correspondences in the following images at the new position. The main purpose of this system was to enable the cart to avoid obstacles when planning the path, but it fulfills all requirements to be a visual odometry. Later, this technique was evaluated by NASA that implemented it in their rovers for the Mars Exploration Program (Cheng, Maimone, and Matthies, 2005). The definition of the name for the meanwhile proven method was set after the publication of Nistér, Naroditsky, and Bergen (2004), who have strongly contributed to the enabling of visual odometry for real-time applications. To understand how a full trajectory of a vehicle's egomotion is estimated just by incrementally measuring distances, in the following a small excursus is given about one of the first implementations in a robot, the wheel-based odometry (Everett, 1995). Finally, the connection between this odometry and the visual odometry in terms of modern computer implementations is established.

Excursus: Wheel Odometry To calculate the trajectory of a wheel-based vehicle using odometry, its kinematic model and the related parameters are needed; some of them are constant and others change over time. As a theoretical example let's consider a differential wheeled robot with 3 degrees of freedom (DOF), which moves in the two dimensional space due to two separately driven wheels with the radius r as described by Borenstein, Feng, and Everett (1996). This robot has non-holonomic constraints,

because it cannot drive directly to every possible point on the two-dimensional plane. For a location that is not directly in line with its orientation, the robot first has to turn and then is able to drive to that location. For simplicity, it is assumed that the time is discrete, otherwise the linear velocity of the wheels and the rotational velocity of the centre would have to be integrated over time.
The pose of the robot at time k is denoted as $\mathbf{P_k} = (x_k \quad y_k \quad \Theta_k)^\mathsf{T}$ (figure 2.8). The two motors each have a wheel encoder that send pulses $m_{L,k}$ and $m_{R,k}$ if the motor rotates. C is the resolution of the encoder that determines how many pulses per full rotation are getting sent and n is the gear ratio of the reduction gear between the motor and the wheel. From equation (2.20) one gets the travel distance $\Delta U_{L,k}$ and $\Delta U_{R,k}$ for the left and right wheel respectively.

$$\Delta U_{L,k/R,k} = \frac{2 \cdot \pi \cdot r}{m_{L,k/R,k} \cdot C \cdot n} \tag{2.20}$$

From the two travel distances $\Delta U_{L,k}$ and $\Delta U_{R,k}$ the linear displacement ΔU_k of the centre between the two wheels can be calculated with equation (2.21a) and the changed heading $\Delta\Theta_k$ can be calculated with equation (2.21b), where b denotes the wheelbase.

$$\Delta U_k = \frac{\Delta U_{R,k} + \Delta U_{L,k}}{2} \tag{2.21a}$$

$$\Delta\Theta_k = \frac{\Delta U_{R,k} - \Delta U_{L,k}}{b} \tag{2.21b}$$

The following equation (2.21) yields the robots position and orientation $\mathbf{P}_{k+1}$ at time $k+1$ from the previous pose $\mathbf{P}_k$ at time k.

$$x_{k+1} = x_k + \Delta U_{k+1} \cos\Theta_{k+1} \tag{2.22a}$$

$$y_{k+1} = y_k + \Delta U_{k+1} \sin\Theta_{k+1} \tag{2.22b}$$

$$\Theta_{k+1} = \Theta_k + \Delta\Theta_{k+1} \tag{2.22c}$$

Similar to this concept, in visual odometry the shift of pixels in consecutive images is used to estimate the motion of the camera. One of the core issues of visual odometry is to find corresponding pixels in two different images that show the same scene. A prerequisite that the two images must meet, is that they have an overlap of same objects in the scene, and that the angle from which they were taken is not too large. At the latest when the angle is 180° in most cases much earlier, although showing the same object, it will not be possible to find correspondences. There are three main systematic approaches to solve this task: feature-based, appearance-based and a hybrid of the two preceding ones. All of this have in common that the goal is to find the hidden model parameters $\mathbf{X}$ from the measurements $\mathbf{Y}$. $\mathbf{X}$ contains the parameters for the camera motion and the model of the world and $\mathbf{Y}$ contains the noisy measurements. The correlation of these two vectors can

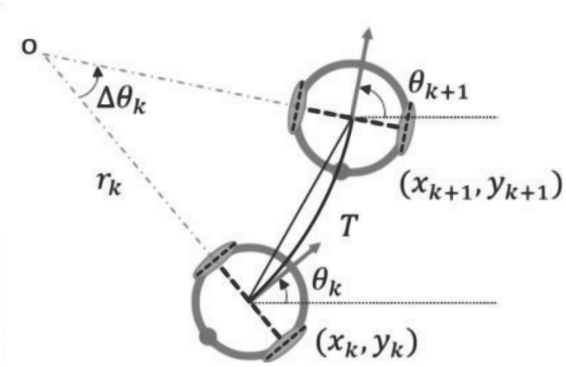


Figure 2.8: Geometrical relation between two successive poses of a differential-wheeled robot (reprinted from Chen et al., 2017).

be expressed in a probabilistic model that estimates the unknown model parameters $\mathbf{X}$ so that the likelihood for perceiving the measurements given those parameters is maximised as represented in equation (2.23).

$$\mathbf{X}^* := \text{argmax}_x \, P(\mathbf{Y} \mid \mathbf{X}) \tag{2.23}$$

2.4.1 Feature-based Method

This method is also referred as *indirect method* because the raw sensor measurement of the pixel is getting processed before it is expressed in the measurement matrix $\mathbf{Y}$. The process involved in this is the feature extraction. To get a feature, the first step is to find key-points in the image. These are points that are distinguishable from the majority of all other points in the image. This applies for instance to primitive geometrical shapes like edges but even more so to corners. After that, the second step is to describe the feature so that it can be found in another image. Finally, the last step is to match the features from two different images.

In general a good feature is based on a key-point that is repeatable like the aforementioned corner but also reliable in terms of uniqueness in the whole image. Corners of repetitive geometrical structures like on a chess-board for example would only meet one of those conditions. Research in this field of detection already started in the 70s with Schmidt Jr. (1971), who implemented an algorithm in the Stanford cart that enabled it to follow a white line on the ground. The later work of his follower **Moravec1980** on the same robot, yielded to one of the first corner detectors. Upon the work of Moravec, Harris, Stephens, et al. (1988) developed a corner-detector that is still used today and a standard detector in the most recent OpenCV library (Bradski, 2000). A modification of the Harris-Corner-Detector is the detector of Shi et al. (1994). Förstner and Gülch (1987) provided a different approach for the detection of corners by using tangential lines. The

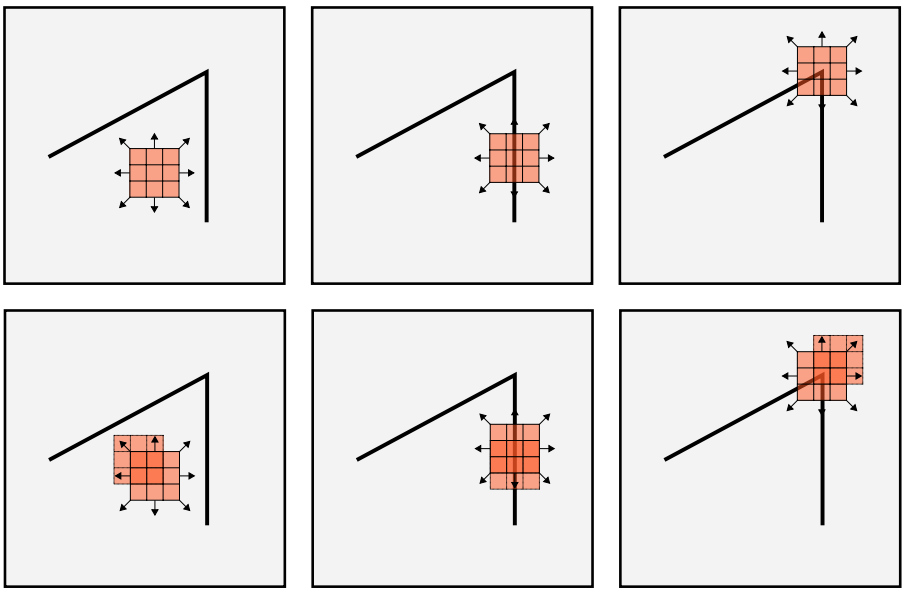


Figure 2.9: Visual explanation of Moravec's corner detector. All six boxes show the same Image I with different positions of the window W. The upper and lower picture each form a pair. The first image pair shows a flat region, where shifting the window does not result in changes of intensities in any direction. Image pair two shows the window on a edge, where only a shift along the line will not change the value. In image pair three the window is on a corner; at this position, a shift in any direction will change the value.

corner is estimated at the point that has the least squared distance to all of the lines. This approach can estimate corners to sub-pixel accuracy and is especially useful for detecting smooth corners.

Feature Keypoint Selection Moravec (1980) proposed a method that compares the measured intensities inside a fixed sized window to the intensities inside that window when it gets shifted into one of eight directions that are evenly distributed in 45° angles around the initial position. The goal is to find a position (x, y) of the window W inside the image that will yield to a large change E for all small shifts $(\Delta x, \Delta y)$ of the window, because this condition applies to corners. Three possible positions for a 3-by-3-pixel window inside an Image I are shown in figure 2.9. For simplicity and without the loss of generality, the method is shown in the following for a two-dimensional greyscale image. In equation (2.24) the sum of squared differences between two window positions is calculated by summing up the squared changes of intensity $I(x_i + \Delta x, y_i + \Delta y) - I(x_i, y_i))$ for all pixels $(n = 9)$ inside the window.

$$E(\Delta x, \Delta y) = \sum_{\substack{i=1 \\ x_i, y_i \in W}}^{n} [I(x_i + \Delta x, y_i + \Delta y) - I(x_i, y_i)]^2 \tag{2.24}$$

Morovec's corner detector is anisotropic, because it uses a box filter which only considers shifts into eight different directions and which assumes that the displacement is in full pixels each ($\Delta x, \Delta v \in \mathbb{Z}$). Harris, Stephens, et al. (1988) improved Morovec's detector by extending it to displacements in all directions and by any small length. If a Gaussian filter is used, the detector becomes isotropic. The term for the intensity change can be approximately linearised for small shifts by a Taylor series expansion up to the first order as shown in equation (2.25).

$$\begin{aligned} I_x &= \frac{\partial I}{\partial x} \\ I_y &= \frac{\partial I}{\partial y} \\ I(x + \Delta x, y + \Delta y) &\approx I(x, y) + I_x \Delta x + I_y \Delta y \end{aligned} \tag{2.25}$$

I_x and I_y are the gradients of the image I at (x, y). Inserted in equation (2.24) this yields to equation (2.26) where the intensities at the initial position of the patch get cancelled out.

$$E(\Delta x, \Delta y) \approx \sum_{\substack{i=1 \\ x_i, y_i \in W}}^{n} [I_{x_i} \Delta x + I_{y_i} \Delta y]^2 \tag{2.26}$$

This can be written in matrix form as shown in equation (2.27).

$$E(\Delta x, \Delta y) \approx \sum_{\substack{i=1 \\ x_i, y_i \in W}}^{n} \left[\begin{pmatrix} I_{x_i} & I_{y_i} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \right]^2 \tag{2.27a}$$

$$= \sum_{\substack{i=1 \\ x_i, y_i \in W}}^{n} \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} \begin{pmatrix} I_{x_i}^2 & I_{x_i} I_{y_i} \\ I_{y_i} I_{x_i} & I_{y_i}^2 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \tag{2.27b}$$

$$= \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} \underbrace{\sum_{\substack{i=1 \\ x_i, y_i \in W}}^{n} \begin{pmatrix} I_{x_i}^2 & I_{x_i} I_{y_i} \\ I_{y_i} I_{x_i} & I_{y_i}^2 \end{pmatrix}}_{\text{structure tensor } \mathbf{M}} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \tag{2.27c}$$

The eigenvalues λ_k and corresponding eigenvectors x_k of $\mathbf{M}$ define the direction respectively the amount of change for E. For the displayed positions in figure 2.9 it means that $\max \lambda_k$ and $\min \lambda_k$ are small for the flat area, $\max \lambda_k \gg \min \lambda_k$ for the edge and for the corner $\max \lambda_k \approx \min \lambda_k$ and both are large as shown in figure 2.10. The response function R presented in equation (2.28) makes use of this circumstance.

$$R = \det(\mathbf{M}) - k \cdot \operatorname{tr}(\mathbf{M})^2 \tag{2.28a}$$

$$= \lambda_1 \lambda_2 - k \cdot (\lambda_1 + \lambda_2)^2 \tag{2.28b}$$

where k is an empirical determined constant which can be adjusted to either achieve higher precision or recall. Normally k is in the range $[0.04, 0.06]$. For a bigger k, the precision is higher with less false positives but at the cost of some missed corner-points; for a smaller k, the recall is higher which leads to more corner-points and also to a higher rate of false positive detections.

From the value of R one can conclude the following:

- if $|R|$ is small, the pixel belongs to a flat area,
- if $R < 0$, the pixel belongs to an edge,
- if R is large, the pixel belongs to a corner.

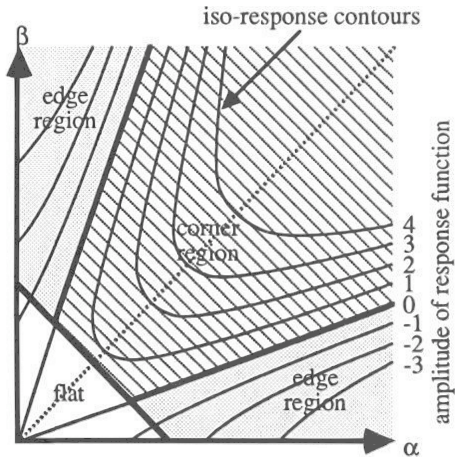


Figure 2.10: Relation between the two eigenvalues λ_1 and λ_2, here depicted as α and β and what statement can be made on the basis of this about the point at which these values were calculated (reprinted from Harris, Stephens, et al., 1988).

Shi et al. (1994) presented a different way to calculate the response by directly evaluating the minimum of the eigenvalues as shown in equation (2.29). It is more efficient in most cases, because no quadratic term needs to be calculated.

$$R = \frac{\det(\mathbf{M})}{\operatorname{tr}(\mathbf{M})} \tag{2.29a}$$

$$= \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} \tag{2.29b}$$

To use this function, only points where the eigenvalue $\min \lambda_k$ is bigger than a certain threshold must be considered. Shi and Tomasi state that this method for tracking corners between two images is superior to the Harris detector in situations where the displacement is not the same for all pixels, but is rather represented by a affine motion vector field. Another corner detector is the *Features from accelerated segment test* (FAST, Rosten and Drummond (2006)), where intensities along a circle around the pixel are getting evaluated as shown in figure 2.11. If at least a certain amount n of adjacent pixels are brighter or darker than the pixel in the centre, the point is considered a corner. A value for n higher than 12 should be chosen according to Rosten and Drummond (2006). This approach is, as the name suggests, faster than the methods presented above, because it only compares values instead of doing matrix multiplications or solving squared terms.

All the aforementioned detectors belong to the category of corner detectors, but there is a second category of detectors, which detects blobs. Blobs are regions in an image that are different from its surrounding in terms of texture, colour and intensity. Instead of using a response function, these methods calculate the *Difference of Gaussian (DoG)* to find keypoints. Figure 2.12 shows how *SIFT (Scale-Invariant Feature Transform)* (Lowe, 2004) - a famous representative of blob detectors - uses the DoG to calculate keypoints.

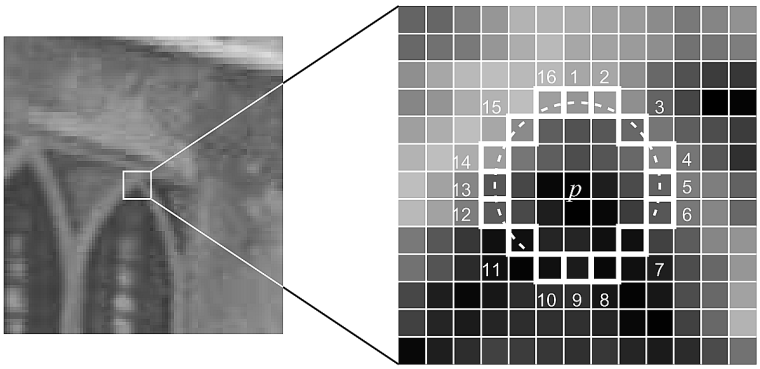


Figure 2.11: The intensities of the 16 pixels along the circle around the point p are getting compared to the intensity of pixel p. The Image shows an example for 12 contiguous pixels that are brighter than the centre (indicated by the dashed line). Therefore, the pixel p indicates a corner (reprinted from Rosten and Drummond, 2006).

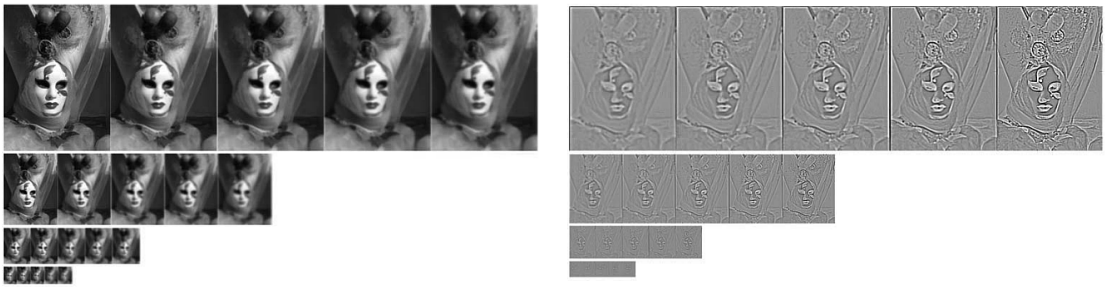


Figure 2.12: The left cluster of images shows the original grey-scale image that was smoothed by Gaussian filters with different sigmas (left to right) and down-sampled each by a factor of 2 per row (from top to bottom). The right cluster of images shows the *difference of Gaussian (DoG)* of two successive Gaussian-smoothed images (reprinted and modified from Siegwart, Nourbakhsh, and Scaramuzza, 2011).

The main advantage of blobs over corners is that they are more distinctive and can be re-detected better after a change in scale. On the other hand, corners yield a better localisation and are faster to compute (Siegwart, Nourbakhsh, and Scaramuzza, 2011). The choice of a suitable detector always depends on the task for which it is to be used and requires thorough consideration. A more detailed comparison of feature detectors in terms of properties and performance can be found in Siegwart, Nourbakhsh, and Scaramuzza (2011).

Feature Description The second step after finding feature keypoints, is to describe them so they can be found in a new image. The simplest way for that is to take the pixels in the patch around the feature and store their intensity and position. The downside of this descriptor is that it is very sensitive to changes in lighting conditions or viewpoint changes. A more robust descriptor, called the Census transform, was presented by Zabih

and Woodfill (1994). Instead of taking the intensities in the patch directly, they are first compared to the intensity of the feature point and depending on whether they are larger or smaller, they are assigned the value 0 or 1. The values are then combined into a binary vector. Another well-proven descriptor is that of SIFT. The SIFT descriptor divides the area around the feature point in 16 boxes of equal size and constructs a histogram of the gradient directions inside every box. The histograms, each with eight bins for every box, are stacked into one vector. The resulting 128-element vector for the feature is normalised to unit length to make it more robust to changes of perspective and lighting conditions. As Fraundorfer and Scaramuzza (2012) state, this concept works well and the "SIFT descriptor proved to be stable against changes in illumination, rotation, and scale, and even up to 60° changes in viewpoint."

There are several other descriptors, of which *Oriented FAST and Rotated BRIEF (ORB)* (Rublee et al., 2011) should be especially emphasised because of its efficiency and wide application (Mur-Artal, Montiel, and Tardos, 2015). Compared to SIFT, ORB is outperformed in terms of accuracy in the most scenarios, but the calculation and matching of features is much faster (Tareen and Saleem, 2018; Karami, Prasad, and Shehata, 2017). As the name suggests, ORB is based on the achievements of FAST and BRIEF (Binary Robust Independent Elementary Features) (Calonder et al., 2010). It calculates the FAST features for a multi-scale image pyramid to make it more robust to changes in scale and then calculates the *intensity centroid* for every corner. The idea behind the intensity centroid is that the geometrical centre of a corner is offset to its intensity centre. The calculation of the intensity centroid after Rosin (1999) is shown in equation (2.30), where the centroid $\mathbf{C}$ is determined from the standard moments m_{pq} of the image intensities $I(x, y)$.

$$m_{pg} = \sum_{x,y} x^p y^q I(x, y) \tag{2.30a}$$

$$\mathbf{C} = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{11}} \right) \tag{2.30b}$$

The orientation of the vector between the corner's centre O and the centroid C can thus be calculated with the quadrant-aware version of arctan, the atan-function equation (2.31).

$$\Theta = \operatorname{atan}(m_{01}, m_{10}) \tag{2.31}$$

With the known orientation, the features can be rotated to a unified direction to make the matching process more robust to rotations.

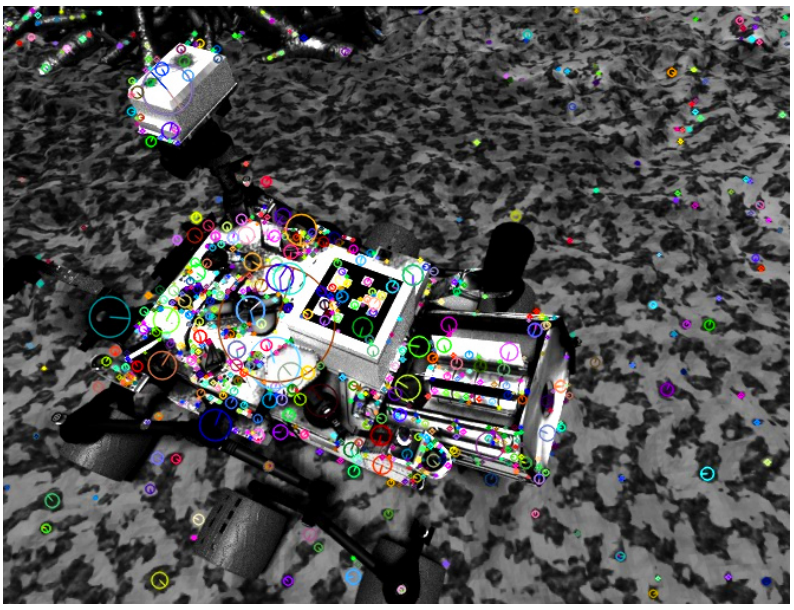


Figure 2.13: The image shows 1000 SIFT feature points with their corresponding descriptor (coloured circle). The descriptor indicates the orientation and scale of the feature.

Feature Matching To match two features, their descriptors have to be compared for similarity. If the descriptor contains the intensity of surrounding pixels, the sum of squared differences (SSD, equation (2.32)) or the normalised cross correlation (NCC, equation (2.33)) between two vectors is a suitable metrics. If the metrics yields a value that is below or above a certain threshold, it can be assume that the two descriptors represent the same point and match them.

$$SSD(x,y) = \sum_{x,y} [f(x,y) - w(x,y)]^2 \tag{2.32}$$

$$NCC(x,y) = \frac{1}{n} \sum_{x,y} \frac{1}{\sigma_f \sigma_w} \cdot f(x,y) \cdot w(x,y) \tag{2.33}$$

where:

$w(x,y)$: the function of the patch around the feature in image I_1
$f(x,y)$: the function of the region in the image I_2 with the same size as $w(x,y)$
n : number of pixels in w and f
σ_w : standard deviation of w
σ_f : standard deviation of f

In case that the intensities have previously undergone a censure transformation, the resulting binary vector is compared with the vector of a feature from another image and if the Hamming distance between those two is below a certain threshold, the features

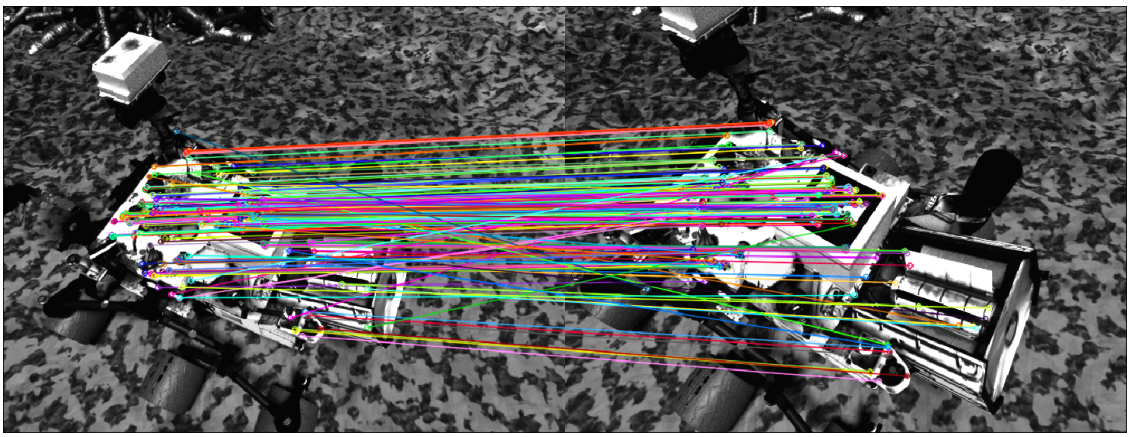


Figure 2.14: Two images of the same rover from different locations with matched ORB-features. Not all matches are correct, which is visible for example through the diagonally crossing lines.

get matched. An example for matched ORB features, which also make use of a binary descriptor, between two images is shown in figure 2.14. The process of matching is usually done by comparing all descriptors of the second image with the first image and merging the features with the most similar descriptors. Since this may result in multiple features from the second image matching one feature from the first image, the check is also done in the opposite direction. Then only those pairs are matched that have each other as the closest solution in both directions. For SIFT, the comparison of the two vectors is performed using the Euclidean distance between them, and the match is only accepted if the distance to the second closest feature is bigger than a certain threshold to inhibit ambiguous assignments.

This method of brute-force matching is very demanding in terms of computational complexity because the matching function f grows quadratic with the number of features n ($f \in \mathcal{O}(n^2)$). The complexity can be reduced by invoking an ordered structure with indices like a search tree which makes use of the arrangement of features in the image and links adjacent ones. Another way is to predict the position of the feature in the new image and search only in a limited area around the predicted area for a matching descriptor. This can either be done by projecting the corresponding three-dimensional object point to the new image with a known transformation to the previous one or, if there is just two-dimensional information from the images and the transformation between them, by looking for the matching feature on the *epipolar line*. An alternative method to separately finding features in two images and matching them is to search the features from the first image in the second image. This approach is known as *tracking* and works best when the transformation of successive images and their discrepancy in for example the lighting conditions is small, so that the feature in the second image is of similar appearance and at a similar position as in the previous image. The *Kanade-Lucas-Tomasi* tracker (Lucas,

Kanade, et al., 1981; Tomasi and Kanade, 1991) is an attempt to make the tracking more robust to changes of the features over long trajectories by applying an affine-distortion model to them.

2.4.2 Appearance-based Method

This method is also referred as *direct method* because the raw sensor measurements of the pixels are directly used as the measurements $\mathbf{Y}$ in equation (2.23). The central characteristic of this method is that it bypasses most of the heuristic feature extraction methods mentioned above, as it takes the intensity measurement of the sensor. This has two main advantages: first, most of the costly computations are not necessary and the direct methods are therefore well-suited for real-time applications on low performance computers. Second, it is possible to use information from the whole image and not just from some distinctive feature points as the comparison of the point selection between the *Direct Sparse Odometry* DSO (Engel, Koltun, and Cremers, 2018b) in figure 2.15 and SIFT in figure 2.13 clearly shows. The previously mentioned KLT-tracker can still keep up with the first advantage but is inferior in terms of combining these two properties in one algorithm (Kerl, Sturm, and Cremers, 2013). However, this does not mean that the KLT is generally inferior to the DSO.
The DSO as well as the *Information-Driven Direct RGB-D Odometry* (ID-RGBDO) (Fontan, Civera, and Triebel, 2020) do not minimise the geometric error of the projected image points as is is done within the algorithms of the feature-based methods. Instead, the photometric error is minimised, which is the difference in intensities between two corresponding points. The new concept of modern direct odometries takes brightness parameters into account while minimising the error, this has the advantage to earlier methods that the approach is more robust to changes in lightning condition and sensor noise. The equation used for the minimisation is shown in equation (2.34)

$$E_{\mathbf{p}j} := \sum_{\mathbf{p}\in\mathcal{N}_p} w_{\mathbf{p}} \left\| (I_j[\mathbf{p}'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\mathbf{p}] - b_i) \right\|_\gamma \tag{2.34}$$

Where I_i is the reference frame and I_j is the current frame that is tracked in relation to I_i. $t_{i,j}$ are the exposure times of the respectively image and $a_{i,j}$ and $b_{i,j}$ are brightness parameters for the frames. $\mathbf{p}$ is a point which is defined by a small patch that includes 8 image points. $\|\cdot\|_\gamma$ is the *Huber norm*, which is a loss function that is more robust in the presence of outliers as the standard squared error loss function. After the minimum is found, the transformation between the images can be calculated with equation (2.35), where $\mathbf{p}'$ is the projected point of $\mathbf{p}$ in $t_{i,j}$.

$$\mathbf{p}' = \prod\nolimits_c \left(\mathbf{R} \prod\nolimits_c^{-1} (\mathbf{p}, d_{\mathbf{p}}) + \mathbf{t} \right) \tag{2.35}$$

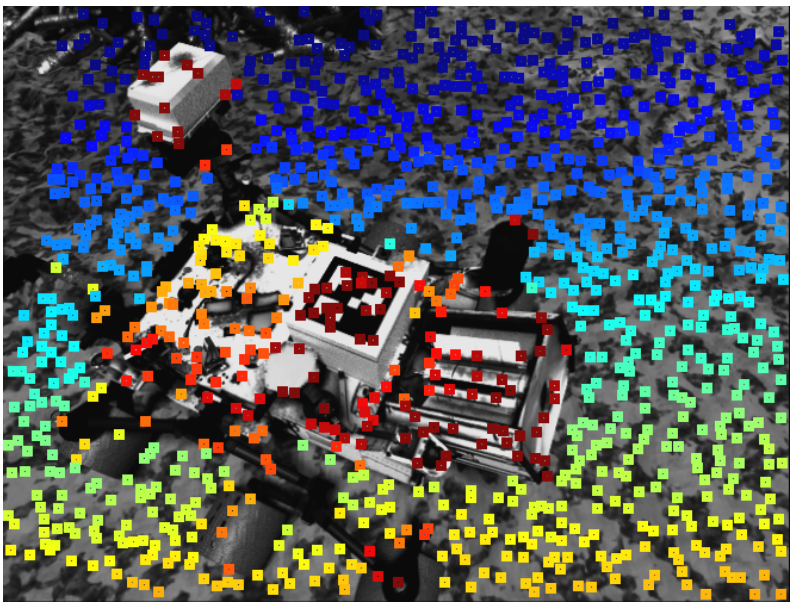


Figure 2.15: The image shows 1000 points that were used by DSO during tracking.

where

$$\begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix} := \mathbf{T}_j \mathbf{T}_i^{-1} \tag{2.36}$$

and $\prod_c$ is the projection of a three-dimensional point to the image plane. $d_{\mathbf{p}}$ is the inverse depth of the point $\mathbf{p}$. As seen in the equations above, the DSO uses a descriptor in shape of a small patch around the measured pixel and relies on the tracking of frames relatively to a keyframe instead of consecutive images. This has the advantage that the drift due to the continuous integration of transformation is kept within limits and the uncertainty has not propagated from the start to the end, but just from keyframe to current frame. The difference of the ID-RGBDO to the DSO is the point selection before minimising equation (2.34). Before solving this equation, the points are selected by their potential contribution to the enhancement of the pose estimation.

This overview of the different methods and their differences will not be deepened at this point, as the basis for a well-founded decision in favour of a method can be made from it, and the further development of a visual odometry is not part of this thesis. For a more detailed explanation of visual odometry, it can be referred to the two tutorials by Scaramuzza and Fraundorfer (2011) and Fraundorfer and Scaramuzza (2012) which were also consulted for this section.

2.5 AprilTag — A fiducial marker

AprilTag is the name of a visual fiducial (lat. *fiducia* - "trust") system that was developed by (Olson, 2011). The system consists of two major parts: a physical part, which is a black and white squared marker and a software part in implementation of a detection and decoding algorithm. The marker that also just by itself is referenced as AprilTag, is shown in figure 2.16. It consists of multiple macro-pixels that encode information similar to the two-dimensional relative of the barcode, the QR-code. The tag is chosen to be black and white due to the high intensity gradient between these two colours; this makes it easier to detect the lines that compose it. A proper detection of the tag and the distinction from the rest of the scene is essential for the further processing. The name *36h11* stands for the *family* of the tag, which means for that example that the tags payload consists of 36 macro-pixels and all tags inside that family have a hamming distance of 11 to each other. The idea behind this has been used for a long time for the calibration of cameras with a checkerboard pattern and was extended here by a coding and a new approach for robust corner estimation. Similar methods have already been carried out before by Abidi and Chandra (1995), for example.

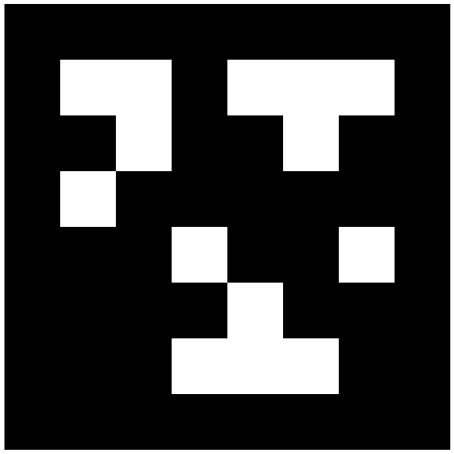


Figure 2.16: AprilTag of family 36h11 with ID 2.

One of the key-ideas behind using this marker system is that there is an object of known size in a scene which can be used to calculate the metric position and orientation of a camera in relation to that object. This can e.g. be used to enhance the performance of an augmented reality scene or - as in the case of this thesis - as a landmark labelling a point of great interest to a UAV. An additional application of the tag is to give orders to robots by showing AprilTags with a specific ID to them; after recognising the tag, the robot will perform a preconfigured task in accordance to that ID. The name of the artificial

marker originates from the *APRIL Robotics Laboratory* at the University of Michigan, in which the system was developed. *APRIL* is an acronym that derives from the topics the laboratory deals with: Autonomy, Perception, Robotics, Interfaces, and Learning. The connection between the physical tag an the algorithm is facilitated by a camera. More precisely an image taken of the AprilTag, placed as an artificial object in the scene, is sent to the detector, which then searches for valid tags that appear in the image. The first part of the detector is an algorithm that finds four-sided regions in the image, which applies, among other things, to the planar tags. In a second step, a decoding algorithm analyses the payload of the region. If it is a valid tag, it should contain a binary encoded ID that is composed by macro-pixels as shown in figure 2.17.
AprilTag is not the first approach that deals with this type of fiducial markers, but has become one of the most widely used in the robotics community, besides ARTag (Fiala, 2005), and was also adapted for NASA's AprilNav (Schuler, Studier, and Bryan, 2018). A predecessor and inspiration to the AprilTag development was ARToolKit (Kato and Billinghurst, 1999), one of the first marker-based systems whose further developments are still used for augmented reality applications. There are several other system that - just to name a few - use circles (Bergamasco et al., 2011) instead of squares or leverage colours and their distinctive representation in an RGB image (Degol, Bretl, and Hoiem, 2017).
AprilTag is now in its third version (Krogius, Haggenmiller, and Olson, 2019) and supports flexible layouts like nested tags or roughly circular shapes. Furthermore, the detection algorithm for the *quads* was improved in terms of increased detection rate, and reduction of the amount of computing time needed for detection (Wang and Olson, 2016). The latter improvement is mainly due to the developments in AprilTag 2.

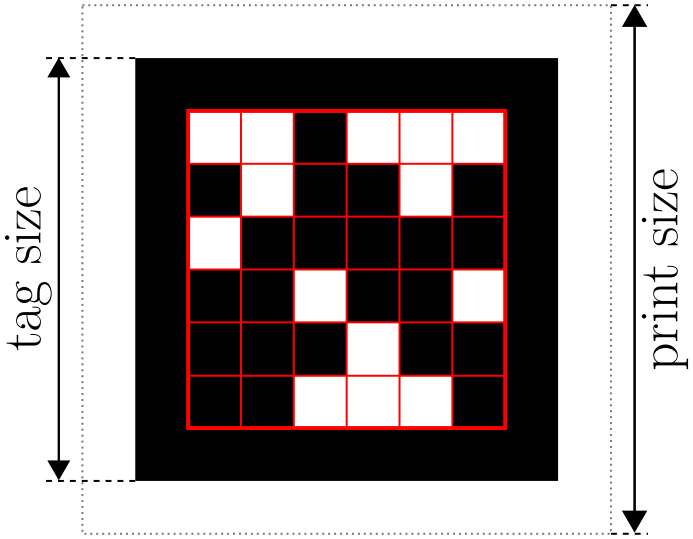
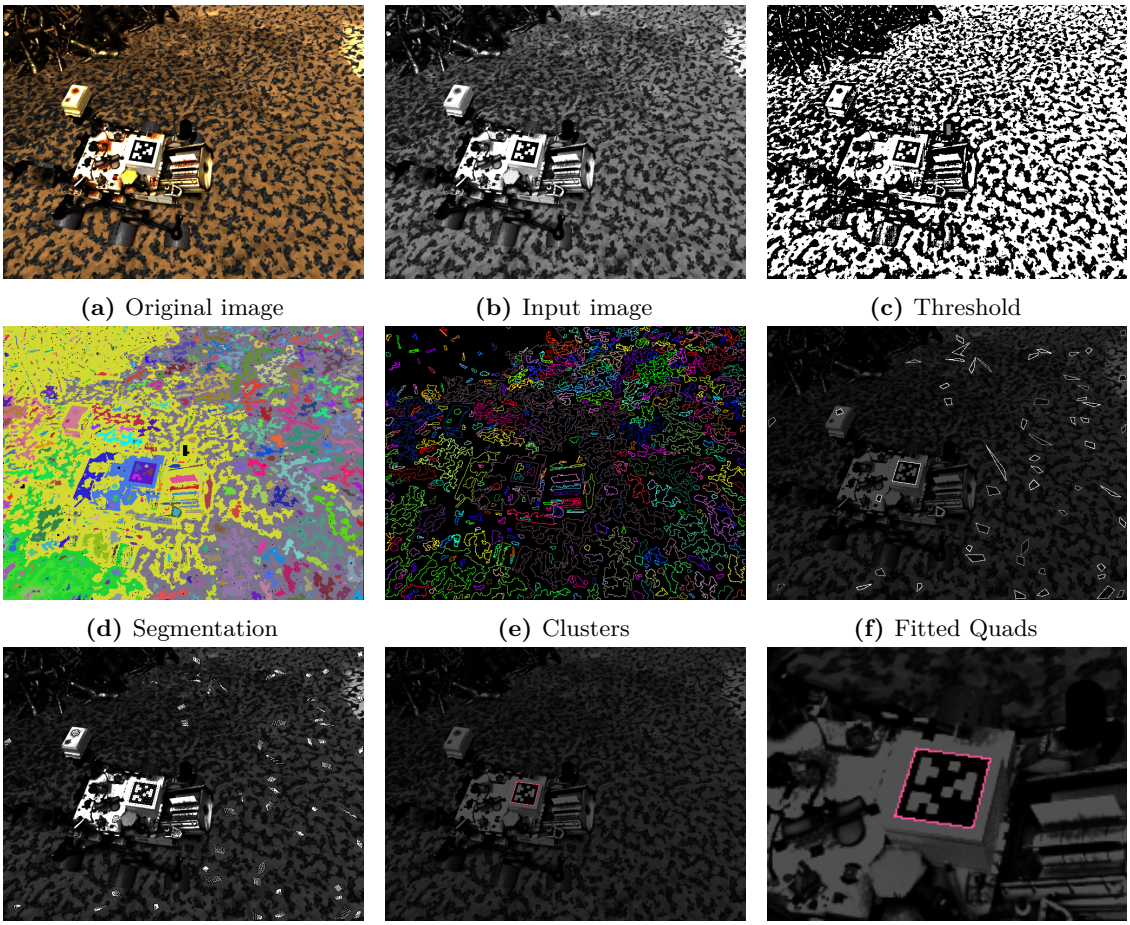


Figure 2.17: The regions of an AprilTag. The actual size (print size) of the tag is larger than the visible part. It contains - from outside to inside - a 1 macro-pixel wide white border, following a 1 macro-pixel wide black border and the data cells (red) around the centre. The tag size is defined by the exterior side of the black border.

Finding an AprilTag The full process of the AprilTag detection with the intermediate steps is shown in figure 2.18. The input image has to be in greyscale, therefore the RGB image has to be converted (figure 2.18b) before it is sent to the detector. Then an adaptive threshold, which depends on the neighbourhood of the pixel, is applied to make it binary (figure 2.18c). The next step is to find edges that have opposite-coloured neighbours and segment them based on of their colour. The connected components are then segmented (figure 2.18d) using the union-find algorithm and the borders of different coloured regions are determined (figure 2.18e). Afterwards, four lines are fit to the points of every border, which will define a *quad* (figure 2.18f). This is done by using the principal component analysis (PCA) (Pearson, 1901) to calculate the best fit of a line through neighbouring points with the same orientation. The corner points are then estimated by the intersection of two lines. The line fitting and corner estimation step is the computationally most expensive part, although it already was reduced in complexity from $\mathcal{O}(n^4)$ to $\mathcal{O}(n)$ for n points by using a precomputed statistics (Wang and Olson, 2016). Not every cluster will yield a valid quad therefore the ones with poor properties are rejected. The valid ones though, are candidates for the decoding algorithm (figure 2.18g) which tries to decode the payload in all four possible orientations. If a valid ID was found inside a quad, the lines and their intersections that define the borders respectively the corners are refined on the original image. Initially those lines were computed using the thresholded image, but by using the image gradients in the new image that are along the edges of the quad, a more accurate line that therefore yields more accurate corner points, can be fitted (figures 2.18h



(a) Original image **(b)** Input image **(c)** Threshold

(d) Segmentation **(e)** Clusters **(f)** Fitted Quads

(g) Decoding **(h)** Detected AprilTag **(i)** Detailed view of (h)

Figure 2.18: The detection process with the main steps. The original image (a) has to be externally converted to greyscale (b) before sending it to the detector. The first step inside is the conversion to a binary image (c) by applying an adaptive threshold. Then connected black or white regions are segmented (d) and a consecutive border around those segments is drawn (e). The next step consists of fitting lines to those borders and constructing so called quads (f). All those quads are candidates for the decoding algorithm that tries to read the tag's payload (g). The final solution is a proper detected AprilTag (h), whose edges are refined using the gradients in the input image (b) again. Image (i) shows a magnified section of (h) to demonstrate the quality of the line fit.

and 2.18i). The edge refinement step improves the accuracy of the pose estimation and was the standard procedure for fitting the quads in the first AprilTag detector before abandoning it in favour of the aforementioned faster method (Wang and Olson, 2016).

2.5.1 Estimating the Pose

The AprilTag is a planar surface and therefore also its corner points. The relation to the camera can therefore be expressed by a planar homography. Using this method, the projection becomes less complex, because of the reduction of the dimension from 3D to 2D. The concept can be illustrated by the relation of the projection matrix **M** and the homography **H** for the case of coplanar points (eqn. (2.37)). The red highlighted elements in the matrices get eliminated because they just add a 0 to every new element of the resulting matrix when multiplying the respective matrices. Note that the projection matrix **M** is a 3×4 matrix which is obtained by multiplying the two matrices above the bracket.

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \underbrace{\begin{pmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{M}} \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} \tag{2.37a}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{2.37b}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{2.37c}$$

assuming $f_x = f_y$ and multiplying the matrices this yields to the homography **H** (eqn. (2.38)).

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fr_{11} & fr_{12} & ft_x \\ fr_{21} & fr_{22} & ft_y \\ r_{31} & r_{32} & t_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{2.38a}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \underbrace{\begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}}_{\mathbf{H}} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{2.38b}$$

The homography is calculated by the *Direct Linear Transformation* (DLT) where the parameters are solved by the following similarity relations between the points (Hartley and Zisserman, 2003).

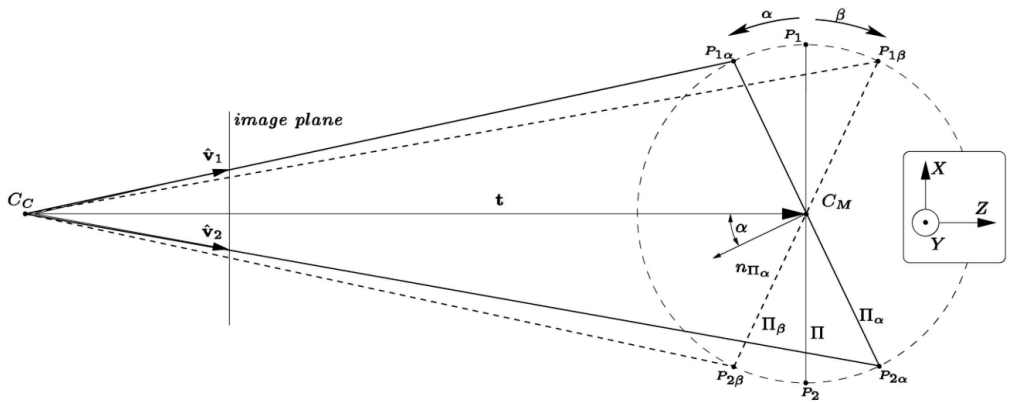
$$x_1 = \frac{h_{11}u_1 + h_{12}u_2 + h_{13}}{h_{31}u_1 + h_{32}u_2 + h_{33}} \tag{2.39a}$$

$$x_2 = \frac{h_{21}u_1 + h_{22}u_2 + h_{23}}{h_{31}u_1 + h_{32}u_2 + h_{33}} \tag{2.39b}$$

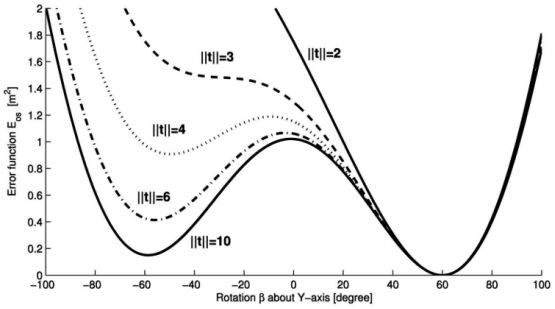
where $\mathbf{x} = (x_1, x_2)$ and $\mathbf{u} = (u_1, u_2)$ are the image- respectively the marker coordinate frame. The missing elements from the rotational matrix can be restored by normalising the two column vectors and then calculating the cross product of them. This can be done because of the orthonormal properties of a rotation matrix. The resulting rotation matrix is then corrected by computing the polar decomposition.
After this step, the payload of the tag is decoded to obtain the identification number. The number is encoded as a binary pattern that is unique in all orientations. As this feature is not relevant in this work, the elaboration on the working principle is not outlined.

2.5.2 Pose-ambiguity Problem

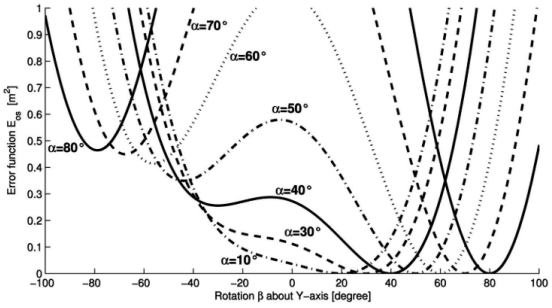
In situations where the projected four corner points are close to each other in the image, the homography cannot be solved unambiguously. This occurs when the tag is far away from the camera, tilted at a large angle or physically small in general. The solution of the error function can then yield a wrong minimum and it is possible that the orientation of the tag is not estimated correctly. However, this affects mainly the rotation and not the translation (Schweighofer and Pinz, 2006), which means that it is of lower importance for the algorithm outlined in this thesis.
The arrangement of the corner points is shown in figure figure 2.19a. Figures 2.19b and 2.19c show the object-space error function for different distances and rotation angles of the tag in respect to the camera. It can be seen that in figure 2.19b the value of the error function for a distance of $\|\mathbf{t}\| = 10$ and an angle $\alpha = 60°$ is also very close to 0 for the negative angle $\alpha = -60°$. With increased noise the solution can become ambiguous and might lead to a wrong pose estimation.



(a) Perspective projection of four corner points.



(b) Distances $\|\mathbf{t}\|$



(c) Rotation angles α

Figure 2.19: The schematic projection of the corner points of the AprilTag to the image plane (a). The object-space error E_{os} for varying viewing distances $\|\mathbf{t}\|$ at a fixed rotation angle of $\alpha = 60°$ (b) and varying rotation angles α (c). All three figures are reprinted from Schweighofer and Pinz, 2006.

2.6 Micro Aerial Vehicle: Ardea

Figure 2.20: The Micro Aerial Vehicle Ardea (Credits: DLR; CC-BY 3.0).

Ardea (Lutz et al., 2020) is a small hexacopter, more specifically a *Micro Aerial Vehicle (MAV)*, which was built by the DLR to take part in collaborative robotic missions and autonomously complete tasks, such as the exploration of uncharted and difficult to access regions (Müller et al., 2018; DLR, 2020a). Ardea is equipped with four cameras that are used for visual odometry (VO) and stereo matching. As declared by Lutz et al. (2020) in their publication about Ardea, the MAV relays on the visual odometry to compensate for the drift that is caused by the integration of the IMU measurements. The VO uses the depth maps from the stereo matching but these maps cannot be calculated when Ardea is standing on the ground. Therefore the state estimation module has to be stopped shortly before the landing. For this thesis, mainly the arrangement and the specifications of the cameras shown in figure 2.21 are of interest that are used for the creation of the simulated dataset in section 3.3.

Figure 2.21: The camera mounting on Ardea (reprinted and modified from Lutz et al., 2020).

Chapter 3

Indirect Tracking of the AprilTag *or* Monocular Visual Odometry with Global Scale

This chapter describes the procedure that leads to the set of solutions presented in this thesis. The goal kept in mind for the realisation, is a successful landing approach, in which it is important to have a high accuracy of the pose estimation as well as an reliable estimation of how certain this estimation is. In order to obtain these estimates at a high frequency, after careful consideration a direct visual odometry was chosen as a framework for indirectly estimating the pose of the AprilTag. The recently developed ID-RGBDO (Fontan, Civera, and Triebel, 2020) is well suited for this purpose, because it uses information theory to keep the points for pose estimation low and thus works very efficiently. The indirect tracking has the advantage that the pose of the AprilTag in respect to the camera can be estimated even if the tag is occluded or not even in the image and detection with the detector would therefore be impossible. Since the ID-RGBDO is a RGB-D based odometry, it receives a dense depth-map (D) and a RGB-image as input, as shown in figure 3.1. The method presented here, involves only a monocular camera with no depth sensor, and as the approach is to remain purely monocular, no additional sensors can be included. Therefore, the only option is to determine the depth values from the RGB images.

For a single image that is possible by placing a planar object with a known size (like the AprilTag) in the scene before taking the picture. Calculating the two-dimensional homography between the object and the image plane, as described in section 2.5, the position of the object in the camera frame can be estimated and the depth for all points on the plane can be calculated by its position and orientation. A second method to obtain a depth image is by using two overlapping images with known transformation between them. The motion stereo algorithm described in section 2.3 can be applied to those images and yields a depth image by triangulating pints that appear in both images. One limitation

Figure 3.1: The ID-RGBDO in- and output as a classical direct RGB-D visual odometry.

that must be taken into account is that the classical rectification approach used in stereo matching cannot be applied here, since the direction of movement is mainly in the direction of the camera z-axis due to the conditions of the camera position and the flight behaviour of the drone. The images were first mapped into spherical coordinates using an already partially existing approach and then the classical search for point correspondences on parallel epipolar lines was performed. In the following, this method and the corresponding implementation in the ID-RGBDO is referenced as a module named *Motion Stereo*.
Figure 3.2 shows the advancements to the ID-RGBDO that were explored and implemented during the thesis, to enable it to work monocular and with proper scale. The figure also serves as a visual guidline for the following explanations. The ID-RGBDO is a apperance-based odometry that estimates the motion between two images by projecting 3D intensity points from a key-frame to the current image and minimising the photometric error of the pixels. The three-dimensional key-frame is obtained by merging the RGB- and depth-image. That means the odometry only needs depth information for the key-frames and then runs a 3D-to-2D motion estimation algorithm.
As a result, the odometry must be able to trigger the Motion Stereo internally if its evaluation of the tracking signals that the current key-frame does not have enough correspondences with the current frame. With the depth image from the Motion Stereo and the current frame, a new key-frame is created. The creation of a key-frame can also be initiated externally by the *Depth from AprilTag* module. It publishes a locally dense depth map of the tags planar surface to the odometry. If this happens, the Motion Stereo is triggered as well and the two depth maps are merged. This merging process marks the point at which the scale is integrated to the odometry. Since the depth values coming from the AprilTag are absolute metric depths and the Motion Stereo only provides depths up to scale, the values of the two maps have to be unified. This is done by a scale factor β for the Motion Stereo depth values which converts them into absolute metric values. The factor β is determined by comparing the depth values located in the area where the tag is projected to the image plane. A more detailed explanation about this algorithm is given in section 3.1.
The next step is the combination of the poses that are published by the odometry and the AprilTag detector. The poses coming from the odometry are in respect to a start position, which is either the origin $\mathbf{0}$ or as in this case a keyframe KF. They get incrementally

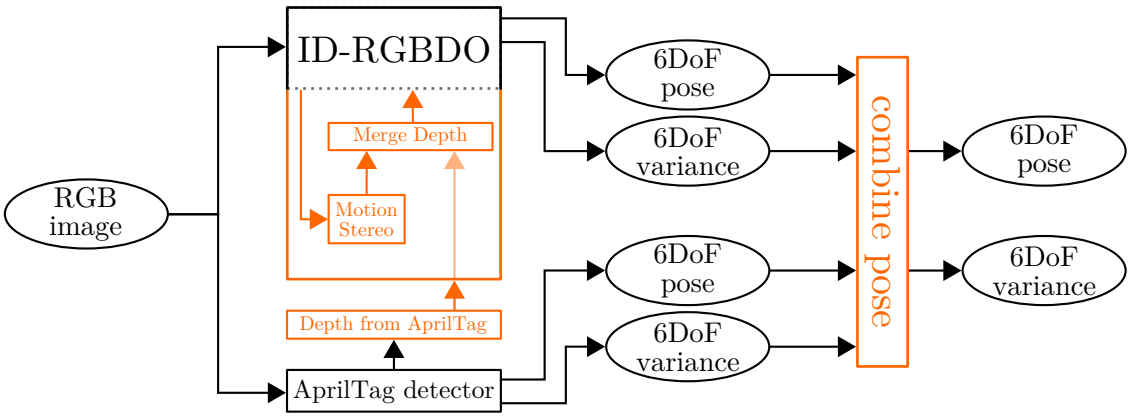


Figure 3.2: The advanced development of the ID-RGBDO to a monocular visual odometry with global scale from an AprilTag and the 6 DoF pose combination module. All changes and extensions are highlighted in red.

integrated and the resulting relative transformation ${}^{KF}\mathbf{T}_f$ is from the current frame f to that starting frame KF. If this transformation is inverted, it yields the pose of the start frame in the current frame. This transformation is then concatenated with the pose from the AprilTag detector to obtain the tag pose in respect to the current frame as shown in equation (3.1). In this way it is possible to estimate the pose of the tag even without detection. Chapter 3 depicts the relation between those three transformations.

$$ {}^{f}\mathbf{T}_{AT} = {}^{KF}\mathbf{T}_{f}^{-1}\ {}^{KF}\mathbf{T}_{AT} = {}^{f}\mathbf{T}_{KF}\ {}^{KF}\mathbf{T}_{AT} \tag{3.1} $$

Their respective uncertainties are also included in the determination of the new absolute pose, by adding the uncertainty of the inverted transformation from the odometry to the initial pose uncertainty of the tag. The error propagation law is used to combine the covariance matrices. This is done by calculating the respectively Jacobian matrices $\mathbf{J}$, which are the first order partial derivatives of the parameters in the transformation matrix and multiplying them from both sides to the covariance matrix that should get propagated. Following the definition of this transformation matrices with errors included, as presented in section 2.1.1, this yields to the following expression in equation (3.2)

$$ {}^{f}\mathbf{\Sigma}_{AT} \ = \ {}^{KF}\mathbf{J}_{f}\ {}^{f}\mathbf{\Sigma}_{KF}\ ({}^{f}\mathbf{J}_{KF})^{\mathsf{T}} \ + \ {}^{f}\mathbf{J}_{KF}\ {}^{KF}\mathbf{\Sigma}_{AT}\ ({}^{f}\mathbf{J}_{KF})^{\mathsf{T}} \tag{3.2} $$

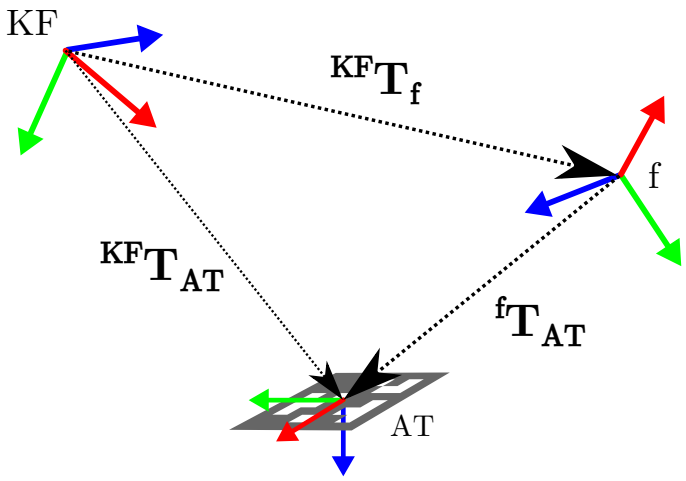


Figure 3.3: The transformations between the different coordinate frames, where KF is the keyframe to which the odometry is performing the 3D-to-2D-matching with the current frame f and AT is the coordinate frame of the AprilTag. The axis are defined as follows: x-axis is red, y-axis is green and z-axis is blue.

3.1 Integration of Motion Stereo and AprilTag into the Odometry

This section describes the practical implementation of the previously elaborated theoretical approach. The explanations depend on the processes shown in figure 3.4 and should be read with reference to this representation. The focus is mainly on the areas and connections marked in orange, since the other parts were used but not analysed in depth. The spherical motion stereo algorithm explained in section 2.3 is implemented in the Motion Stereo module which is part of the ID-RGBDO-MSAT (ID-RGBDO MotionStereo-AprilTag) as shown in figure 3.4. The creation of a keyframe with this method is shown as pseudocode in algorithm 1.

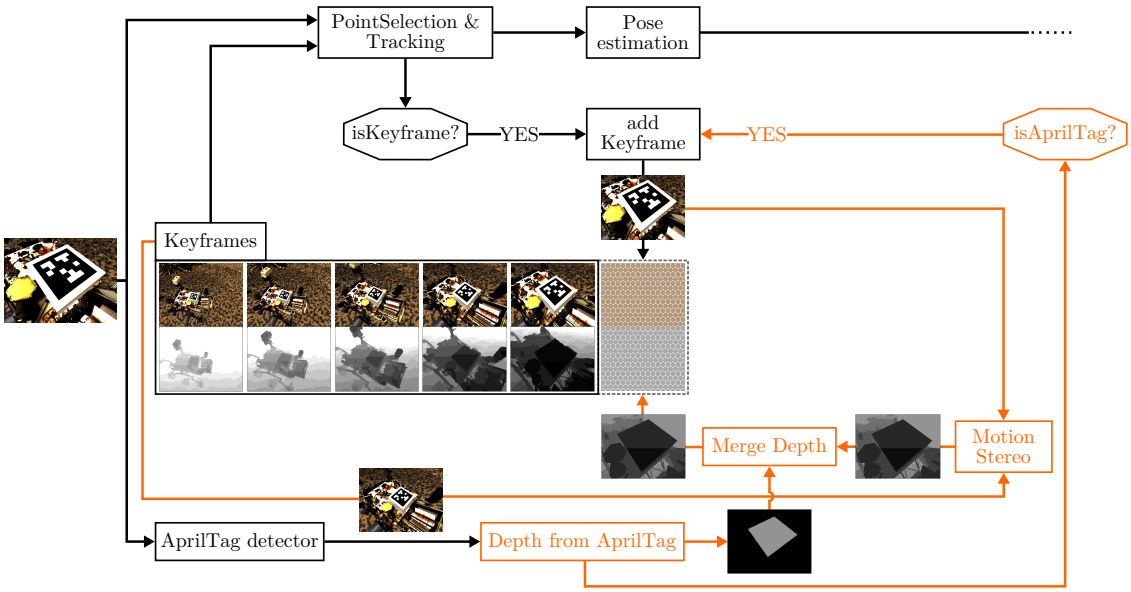


Figure 3.4: Detailed layout of the ID-RGBDO-MSAT.

It is a central element for realising the generation of a keyframe by providing a dense depth map. The Motion Stereo receives two greyscale images. One of them is the most recent frame and the other is one of the key-frames which is selected by the best suiting transformation between it and the current frame. The selection mechanism calculates the transformation between the current frame and all keyframes (in this thesis, 7 keyframes were stored) and returns the image that has a ratio higher than $\frac{1}{30}$ and lower than 1 between the translation of the frame and the median depth of the last generated depth map.

This value turned out to yield good results and is also geometrical funded because the measured disparity for far away objects gets smaller if the baseline decreases and if the movement close to the ground is too large, there cannot be found many correspondences in the images. However, if no keyframe fulfils this requirement, the fallback is to use the previous one. While this procedure allows to estimate the ego-motion of the camera up to scale, it is not the application in this work. It can be seen as an additional property that was established during the work on the whole project.

Besides the internal triggering of the keyframe creation, there is a second — and more relevant — way of initialising this process. If an AprilTag is detected, the *Depth from AprilTag* module calculates a locally dense depth map of the tags planar surface and triggers the *addKeyframe* function inside the ID-RGBDO, which starts the operational sequence described above. The difference this time, is that the *Merge Depth* module does not simply forward the depth image D_{MS} of the motion stereo, but uses the additional information of the AprilTag depth image D_{AT} and merges both images. This is done by creating a binary mask from D_{AT} and applying it to the D_{MS}. This process exposes the

areas of D_{MS} in which the depth values estimated for the area of the marker are located. The median is calculated from the points in this cropped image D^*_{MS} and the points in D_{AT}. The quotient of the median of D_{AT} and D^*_{MS} is used to determine the scaling factor s for D_{MS} (eqn. (3.3)).

$$s = \frac{median(D_{AT})}{median(D^*_{MS})} \tag{3.3}$$

The median is used instead of the mean, because it is more robust to outliers which occur commonly in the depth image estimated by motion stereo if the epipole is close to the marker.

The depth map from the AprilTag detection is calculated by constructing a plane from

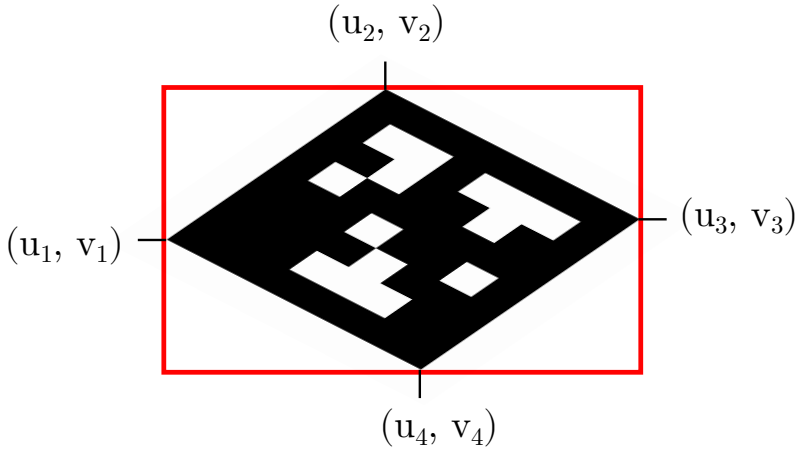


Figure 3.5: The four corner points (u_i, v_i) restrict the area for possible marker points.

the normal vector **n** and the support vector **p**. The support vector corresponds to the estimated translation between the camera and the marker and the normal vector is the orientation of the markers z-axis to the camera frame. The corner points of the marker in the image define the sector in which all image points (and also some points outside) of the tag are located as shown in figure 3.5. All the points inside the rectangular area are then checked for their position in relation to the lines that define the markers edges. If a point is inside, the line from the camera centre to it is used to calculate the puncture point with the plane. This yields the depth value for the map.

The ID-RGBDO is a software that is still being developed; therefore some changes had to be made in its internal processes, which are not discussed further due to the lack of methodological relevance for this work.

Algorithm 1 ID-RGBDO-MSAT algorithm - setting a keyframe

Input: sequence of frames I, keyframe stack KFs ▷ AprilTag must be present
Output: a $keyframe$ (corresponding greyscale- and depth-image)

```
for-loop represents progressing time and frame is the most recent image:
for frame in I do
    procedure AprilTagProcessing(frame, Specs_AT)
        function AprilTagDetector(frame)                    ▷ external program
            Runs on frame, detects visible AprilTags and calculates the spatial
            transformation between the AprilTag and the camera + the ID of the
            tag. The specifications Specs_AT of the physical tag have to be known.
            return ^cam T_AT, ID_AT      ▷ AprilTag in the cameras coordinate frame
        end function

        if AprilTag was found then
            function DepthFromAprilTag(^cam T_AT, Specs_AT)
                1. define a rectangular area A_0 around the tag using the 4 corner points
                2. check which point in A_0 is also inside the tag area A_AT:
                for point P_{0,m} inside A_0 do
                    if P_{0,m} is inside A_AT then
                        calculate 3D location (x, y, z)_m in camera frame of point P_{0,m} at
                        location (u, v) in image frame
                        get distance in z-direction from (x, y, z)_m
                        NOTE: not the euclidean but the z-distance is used
                        depth_AT(u, v) ← z          ▷ z-value is stored at location (u, v)
                    end if
                end for
                return depth_AT
            end function
            isAprilTag ← true
        end if

    end procedure
```

```
28:   procedure TRACKING(frame)                ▷ unchanged procedure of the ID-RGBDO
29:
30:     if ISKEYFRAME(frame) or ISAPRILTAG(frame) then
31:       function ADDKEYFRAME(KFs, frame)
32:
33:         search for second frame to run MOTIONSTEREO:
34:         function SEARCHSUITINGFRAME(KFs, frame)
35:           for KF_i in KFs do
36:             calculate ^{KF_i}T_{frame}    ▷ spatial transformation between frames
37:             if minValue ≤^{KF_i} T_{frame} ≤ maxValue then
38:               return KF_i                  ▷ matching keyframe from KFs
39:             end if
40:           end for
41:         end function
42:
43:         get a depth image from the two frames:
44:         function MOTIONSTEREO(KF_i, frame, ^{KF_i}T_{frame})
45:           1. rectify KF_i and frame with known transformation ^{KF_i}T_{frame}
46:           2. calculate disparity d_i for corresponding pixels
47:           3. calculate depth_MS from the disparities d_i with stereo matching
48:           return depth_MS
49:         end function
50:
51:         merge the two depth images:
52:         function TAGMOTIONDEPTH(depth_MS, depth_AT)
53:           depthAT comes from the APRILTAGPROCESSING procedure
54:           1. compare amount of pixels P_j in the area of the AprilTag in both
              depth images
55:           if Σ_{j=1} P_j^MS / Σ_{j=1} P_j^AT > 0.1 then
56:             2. calculate median M of all depth values in both images:
57:             3. calculate factor F for adjustment of the depth values in depth_MS:
58:             F ← M^AT / M^MS
59:           else
60:             F ← 1
61:           end if
62:           return depth_MERGED ← depth_MS · F
63:         end function
64:
65:         return newKeyframe ← [depth_MERGED, frame]
66:       end function
67:     end if
68:
69:   end procedure
70: end for
```

3.2 Error of AprilTag detections

An evaluation of the AprilTag detector system has already been done by several people during the last decade (Vetter, 2015; Nissler et al., 2016; López-Cerón and Cañas, 2016; Sagitov et al., 2017; Liang et al., 2018; Abbas et al., 2019; Kunz et al., 2019). Vetter (2015) proposed a empirical error estimation by shifting each of the corner points by 1 px in 4 directions, which results in $4^4 = 256$ possible combinations for a rectangle. He then estimates the pose for every combination using a *PnP* solver and takes the largest deviation from the arithmetic mean as error. In addition to that approach, he offers another heuristics in shape of a precomputed look-up table in which the errors of 40000 AprilTag detections from different angles and distances are stored. Those values were obtained by a simulation. The results of this error model applied to a real-world SLAM problem are shown in table 3.1. The part of his work which is important for this thesis is the the quadratic error model. In table 3.1 the error of that model and the empirical model are highlighted for comparison. What emerges from this comparison is that both models give similar estimates of the error for that scenario. The model, which makes use of the online computation of the 256 possible corner points that emerge from shifting the detected points, is much worse, than the other methods. This is not surprising, since the assumed inaccuracy in detection was chosen too large. The corner points of the AprilTag are estimated in principle analogous to the procedure for a camera calibration and are defined by the intersection of two straight lines. Hartley and Zisserman (2003) states that an error of less than $\frac{1}{10}$px is achieved if points are measured that way.
The geometric error in the measurement of the corner point and the geometric error in the object point derived from it have the following linear relation $w \cdot d = f\Delta$ (figure 3.6). The effect of this error on the marker pose, although is not linear. There are analytical methods that solve for the uncertainty of planar homographies (Negahdaripour, Prados, and Garcia, 2005), but an analytical estimation of the combined error for the whole detection process would go beyond the capabilities of this work. Instead of considering each source of error (fig. 3.7) individually and setting up a complex model, an error estimation is used here that has proven itself in other works.
Schuster (2019), who adopted Vetters model in his work, endorses that the heuristic error model can be simplified to a model where (co-)variances scale quadratic with the distance. However, this simplification has one disadvantage: positions in which a pose ambiguity is likely to occur are underestimated and a significantly too small error is indicated. This issue is not so relevant for the task presented here as an pose ambiguity occurs only at a far distance to the tag and not at close range, when a good pose estimation of the landing spot is relevant for the orientation of the MAV. Another examination of the AprilTag and its accuracy was done by Kunz et al. (2019), who investigated its use in the context of medical applications. He proposed to use AprilTags to track and estimate

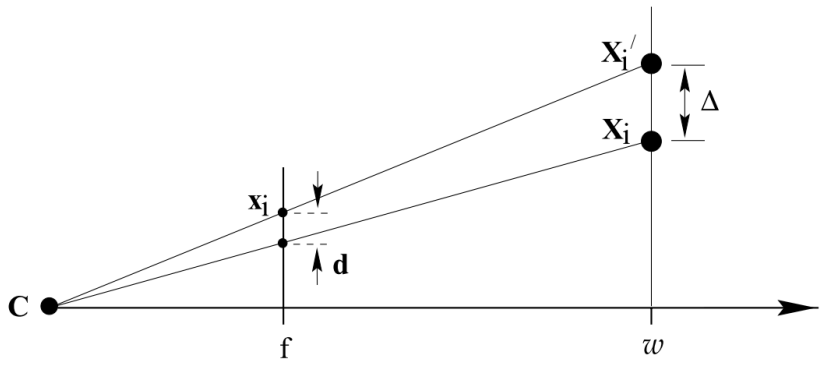


Figure 3.6: The linear relation between the error d of the corner point $\mathbf{x}_i$ and the resulting error Δ of the object point $\mathbf{X}_i$ at a distance $z = \omega$ (reprinted from Hartley and Zisserman, 2003).

the position of patients during computer assisted surgeries. The hardware used for this scenario exceeds the capabilities of a MAV in terms of resolution and computational power, but the cause of the error and the resulting error estimation is similar in both applications. Kunz concluded that the marker size required to determine the 6 DoF pose with constant accuracy increases linearly with the distance to the camera. As the size of the marker is defined by the length of its edge, the area of the tag increases quadratic when doubling the size; thus he comes to a similar conclusion as Vetter (2015) and Schuster (2019). Equation (3.4) was therefore modified from the one mentioned by Schuster, because Schuster just takes the distance d into account; instead of the ratio between distance and tag size l as Kunz points out.

$$\Sigma_{quadratic} = \Sigma_{constant}(1 + \frac{d}{l})^2 \tag{3.4}$$

where:

$$\Sigma_{constant} = 10^{-6} \cdot \begin{pmatrix} 25 & 0 & 0 & 0 & 0 & 0 \\ 0 & 25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix} \tag{3.5}$$

The values for the standard deviation of the rotation and translation are also inspired by Schuster and were just slightly adjusted after some comparison of their consistency with the empirical data obtained by the synthetic dataset. The initial values of Schuster were chosen that high, so that they also include the ambiguities but therefore yield an overestimated uncertainty for all detection at a close range. The variances shown in equation (3.5) have the units cm^2 for the distance and rad^2 for the angles and correspond

Table 3.1: Total error of 3D positions for SLAM trajectories (length: 106 m using different error models for the AprilTag detection of tags and computation of pose estimations. The tags were used as static landmarks. All values taken from Vetter (2015); the empirical error model is denoted as EEM.

	error [m]		
Estimation method	**max.**	**avg.**	**RMS**
constant model	1.26	0.35	0.48
quadratic model	**0.34**	**0.15**	**0.17**
EEM (online only)	8.49	1.06	1.96
EEM (pre-computed only)	0.33	0.13	0.14
EEM (online + pre-computed)	**0.33**	**0.13**	**0.14**

to a standard deviation of 0.5 cm respectively 0.1° when converted. This values would have to be changed if another camera is used, because they also depend on the focal length and camera resolution.

Jin, Matikainen, and Srinivasa (2017) also examined the AprilTags performance and the effect of detection errors on the pose estimations by simulating 10000 detections. He came to the conclusion that the rotational error is much higher relative to the absolute value than the translation error.

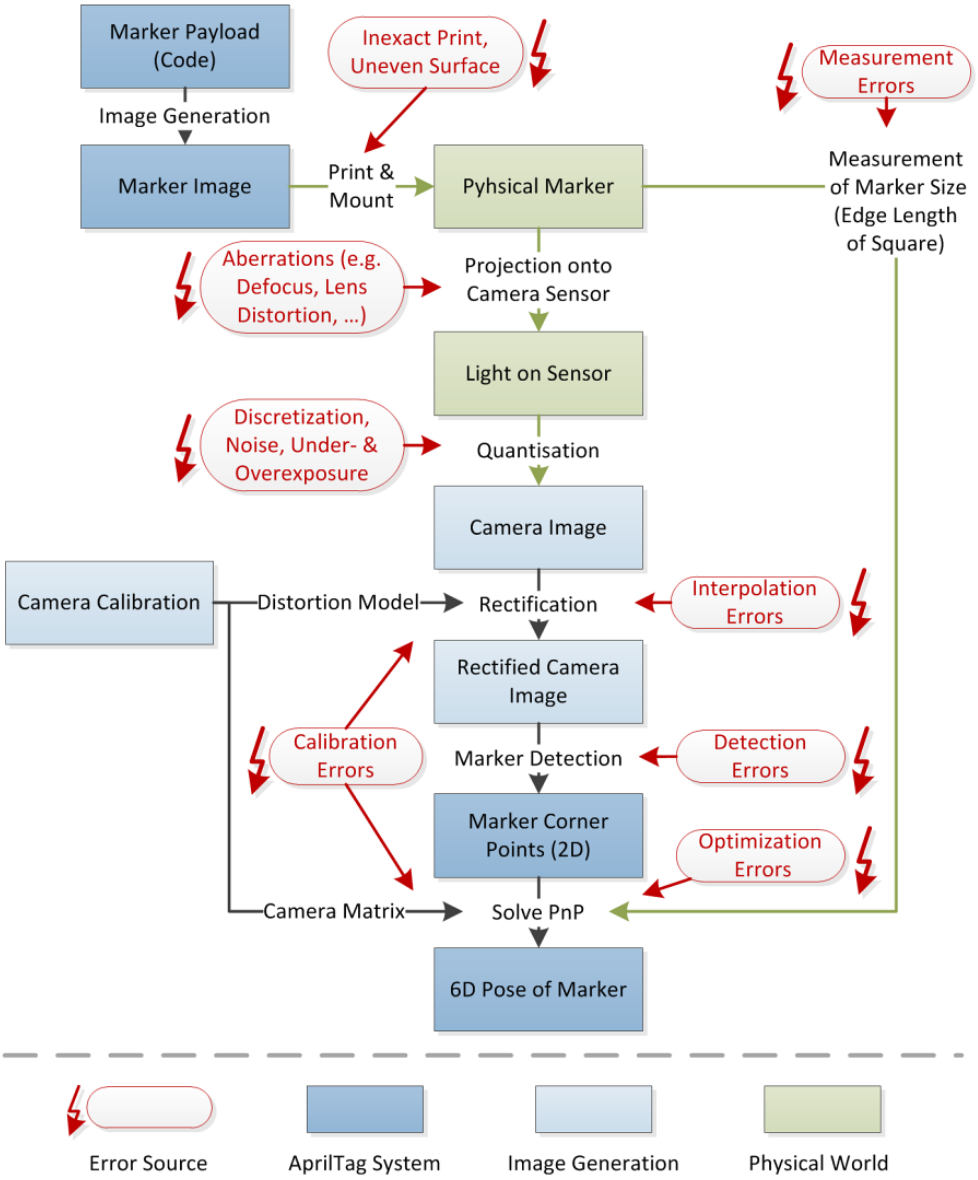


Figure 3.7: Overview of the AprilTag detection system and the possible errors that may occur during the process (reprinted from Schuster, 2019).

3.3 Simulated Dataset

For testing the concept and evaluating its performance, a synthetic dataset was created by using the open-source 3D computer graphics software Blender 2.82 (Blender Online Community, 2020). The scenario is inspired by a exploratory flight of the MAV Ardea in a barren desert and models a potential manoeuvre for landing on a rover. The rover used in the scene is an already existing 3D model of NASAs *Curiosity* rover (JPL). The parameters of the virtual camera were also adapted from one of Ardea's four cameras; the one facing downwards with an angle of 30° between the z-axis of the camera and the vertical axis of the body-frame. This camera was chosen, because it allows to detect the AprilTag from further away even if the MAV is flying at a low altitude and is therefore best suited for this landing approach. An overview of the scene and a close-up to the rover, which has a size of 2.9 m × 2.7 m × 2.2 m, is shown in figure 3.8. The relevant camera parameters can be seen in table 3.2. The duration of the flight is 53.3 s and consist of 640 RGB images, with corresponding depth maps. All images were blurred with a gaussian point spread function of kernel size 3×3 to simulate motion blur and off-focus. Ground truth data was generated both for the absolute pose of the camera in the world frame and for the relative pose of the camera to the AprilTag.

Figure 3.8: Overview of the scene (left) and a detailed view of the rover with the AprilTag on top (right), which marks the MAVs landing spot.

Table 3.2: The parameters of the virtual camera used in Blender.

Parameter		**Value**
focal length:	f_x	396.79 px
	f_y	396.79 px
principle point:	p_x	333.00 px
	p_y	253.00 px
resolution:	w	666.00 px
	h	506.00 px
field of view:	α_h	80.01°
	α_v	65.04°

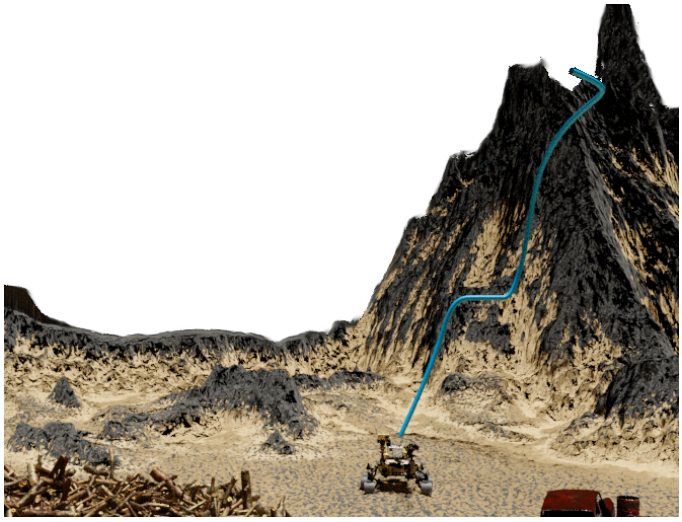


Figure 3.9: The 42.52 m trajectory (blue) of the MAV from frame 300 to 640 in spatial view.

Chapter 4

Results

In this chapter, the performance of the ID-RGBDO-MSAT is evaluated on the synthetic dataset. To provide a basis for a comparison, first the output of the AprilTag detector as well as the ID-RGBDO in its pure form is compared with the ground truth data. Afterwards, the new method is compared to that results and the ground truth. All visual odometry measurements of the ID-RGBDO or rather the ID-RGBDO-MSAT were taken by tracking 256 points in between consecutive images. The definition of the angles follow the following definition: Ψ is a rotation about the x-axis (*pitch*), Θ is a rotation about the y-axis (*yaw*) and Φ is a rotation about the z-axis (*roll*).

4.1 Pose Estimations for separate execution of the AprilTag detector and ID-RGBDO

In this chapter, the results from the aforementioned approaches are presented. They were obtained with the simulated dataset. The sometimes different scaling of the y-axis in the figures must be taken into account when comparing the measured values.
In figures 4.1 and 4.2 the AprilTag detector is compared to the ground truth of the AprilTag and in figure 4.3 the difference between the respective parameters is shown. The section depicted in the diagrams does not show the whole trajectory, because the first tag detection occurred at frame 437. Especially in figure 4.3, the deviation of the detector from the ground truth in the beginning of the section can be seen. The estimations from the AprilTag detector approaching the true values over time and the estimation error in the last 20 frames is less than 1 cm, which is apparent in the magnified view on the right side of the figure. For the frames around 500 there is no tag detection, because the camera is panned back to simulate a disturbance during which the view to the tag is interrupted. In figure 4.2 in the second diagram on the left side, the pose ambiguity is clearly visible, as Θ has a variation of more than twice the nominal angle value during the first detections. The pose ambiguity can also be seen in the plots of the other two

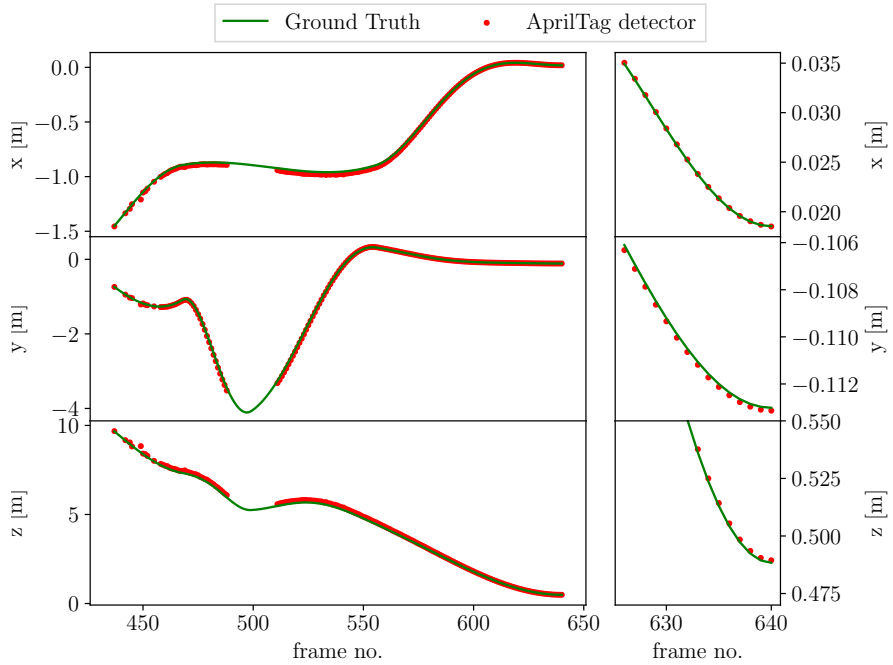


Figure 4.1: The accuracy in location of the AprilTag detector estimations compared to the ground truth. The different scaling of the y-axis must be taken into account when comparing the measured values.

angle estimations Ψ and Φ, albeit not as prominent as for Θ. The position estimation in figure 4.1 is not affected by this ambiguity, because the centre of the tag stays at almost the same position for both solutions. The estimated values of the angles approach the true values over time and the estimation errors are smaller than a tenth of a degree in the last frames. The frequency of detections shown in the aforementioned figures does not represent the frequency at which the detector runs in the setup with the ID-RGBDO and was just run on all images for the purpose of evaluation.

Figure 4.4 shows the estimation errors of the ID-RGBDO when it is provided the simulated depth maps as input. The estimations of the position are closer than 4 cm to the true value for all three directions during the whole experiment. The errors of the angular estimation of Φ rises to a maximum of $\Delta\Phi = 0.18°$ in the last part of the test, where the camera is close to the ground and there is a rotation around its z-axis. The estimation errors of Θ decrease for that section, which most likely is just a coincidence and not a correlation. Analogues figures to those of the previously presented absolute position and orientation measurements of the AprilTag detector, but for the ID-RGBDO can be found in the appendix.

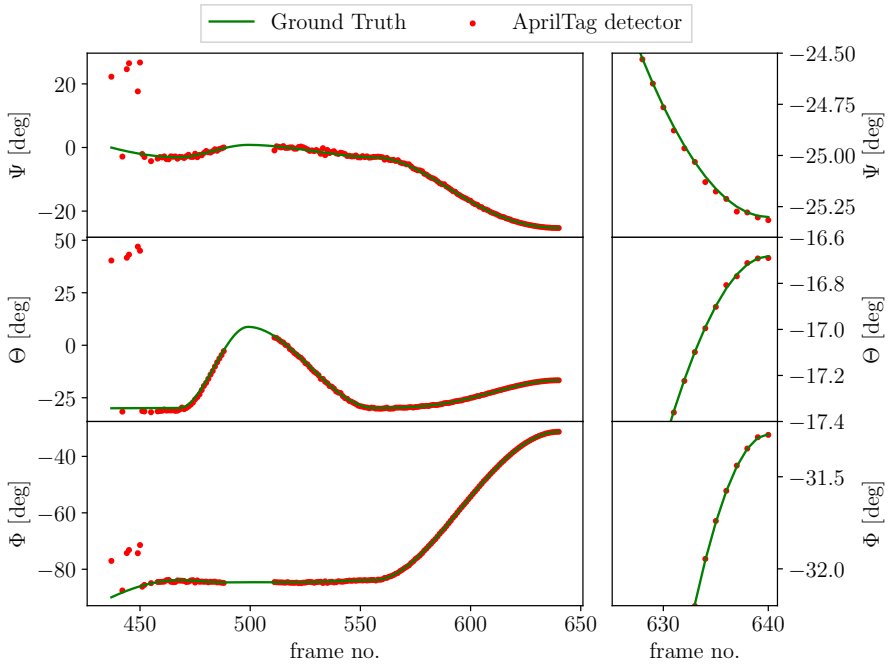


Figure 4.2: The accuracy in rotation of the AprilTag detector estimations compared to the ground truth. The different scaling of the y-axis must be taken into account when comparing the measured values.

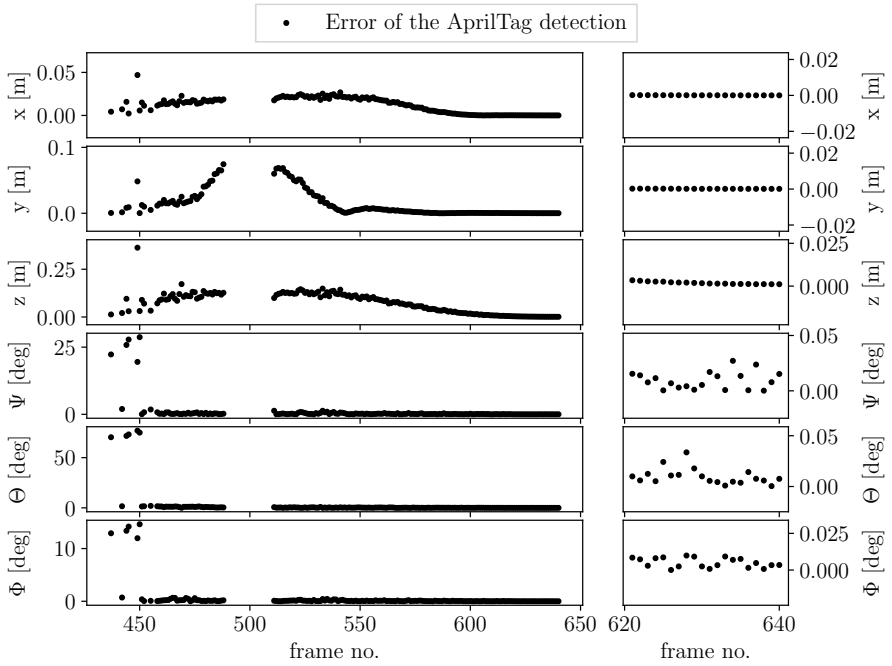


Figure 4.3: The estimation error in rotation and location of the AprilTag detector. The different scaling of the y-axis must be taken into account when comparing the measured values.

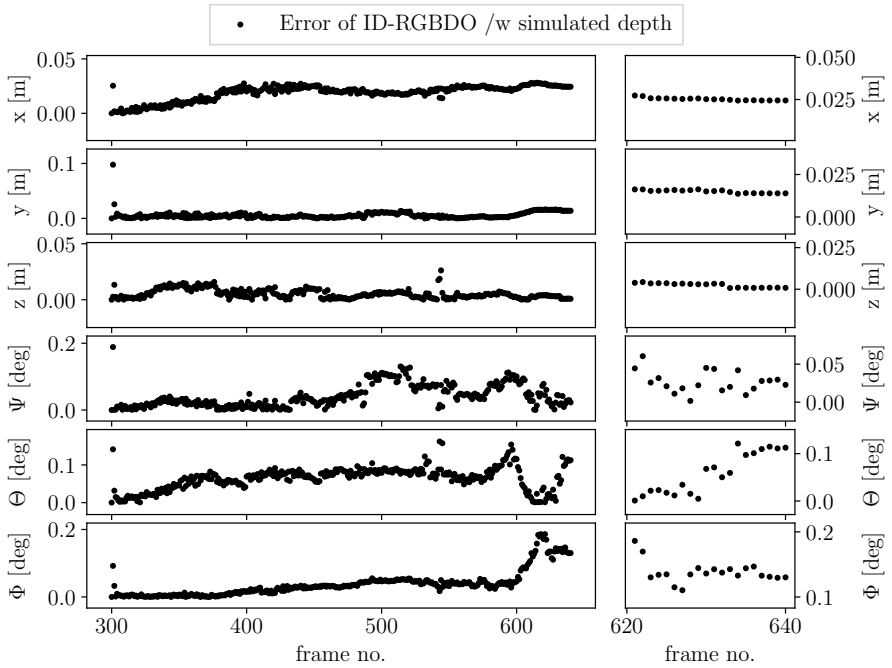


Figure 4.4: The estimation error in rotation and location of the ID-RGBDO estimations when running with simulated depth maps. The different scaling of the y-axis must be taken into account when comparing the measured values.

4.2 Pose Estimation with the ID-RGBDO-MSAT

For the interaction of the AprilTag detector, the ID-RGBDO and the Motion Stereo, two different methods were evaluated and compared, which differ in the initialisation of the ID-RGBDO-MSAT. The pose estimations in figures 4.5 to 4.7 show the results of the first approach, where the initialisation of the ID-RGBDO-MSAT was done with one simulated depth image to generate the first keyframe with an absolute scale. As seen in figure 4.7 the errors rise over time, which is due to the consecutively integration of poses. The error for the translation in z-direction rises to a maximum of $\Delta z = 25.4\,\text{cm}$ just before the end of the trajectory.

Figures 4.8 to 4.10 show the outcome of the second approach, where the ID-RGBDO-MSAT receives no external information about the scale and estimates the depth map for the first keyframe from stereo matching with frame 280. The transformation and the depth image are only estimated up to scale and therefore the trajectory is not scaled properly. It is just a coincidence that the trajectory is not further away from the ground truth, because the baseline is normalised to 1 when no absolute transformation between the two images is known. That means, the real translation between frame 280 and frame 300 is larger than 1 m but not by too much.

At frame 437 the first measurement of an AprilTag is taken into account and the depth

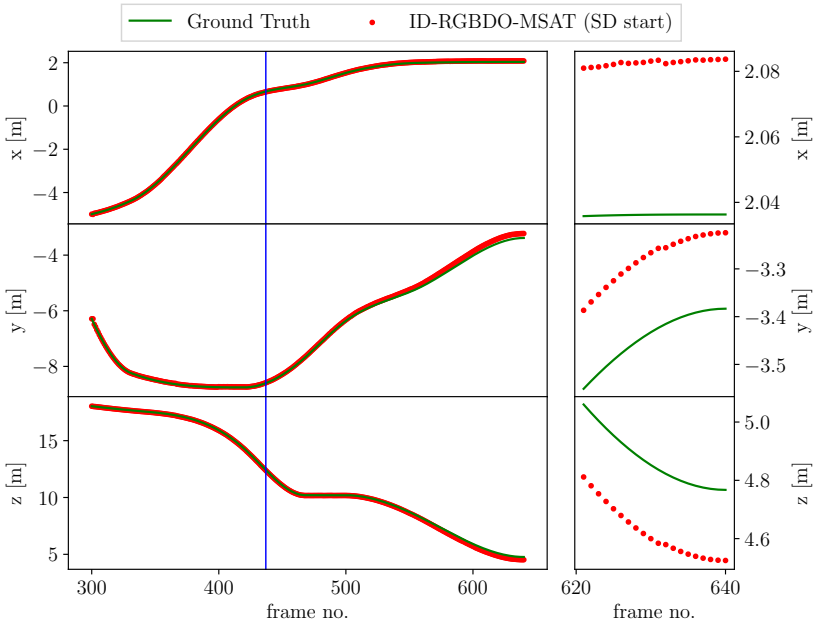


Figure 4.5: The accuracy in location of the ID-RGBDO-MSAT estimations when running with depth maps estimated by motion stereo compared to the ground truth. The odometry was initialised with one simulated depth map. The first detection of an AprilTag is marked with a vertical blue line. The different scaling of the y-axis must be taken into account when comparing the measured values.

images estimated by motion stereo are scaled. This effect is clearly visible in the location errors shown in figure 4.10. After the tag is involved into the visual odometry, the error does not get much bigger till the end of the scene, which can be seen by the parallelism of the trajectories after the detection. The angles are as expected not affected by the wrong scale, which can be seen in figures 4.8 and 4.9. For this approach, the internal bundle adjustment of the ID-RGBDO-MSAT was not used, because it would cause irrational adjustments when combinig the differently scaled depths.

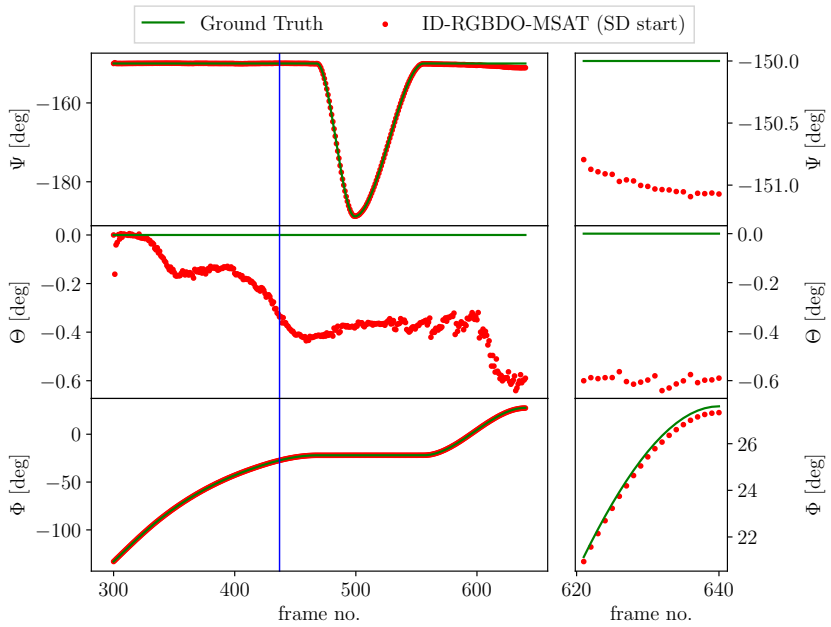


Figure 4.6: The accuracy in rotation of the ID-RGBDO-MSAT estimations when running with depth maps estimated by motion stereo compared to the ground truth. The odometry was initialised with one simulated depth map. The first detection of an AprilTag is marked with a vertical blue line. The different scaling of the y-axis must be taken into account when comparing the measured values.

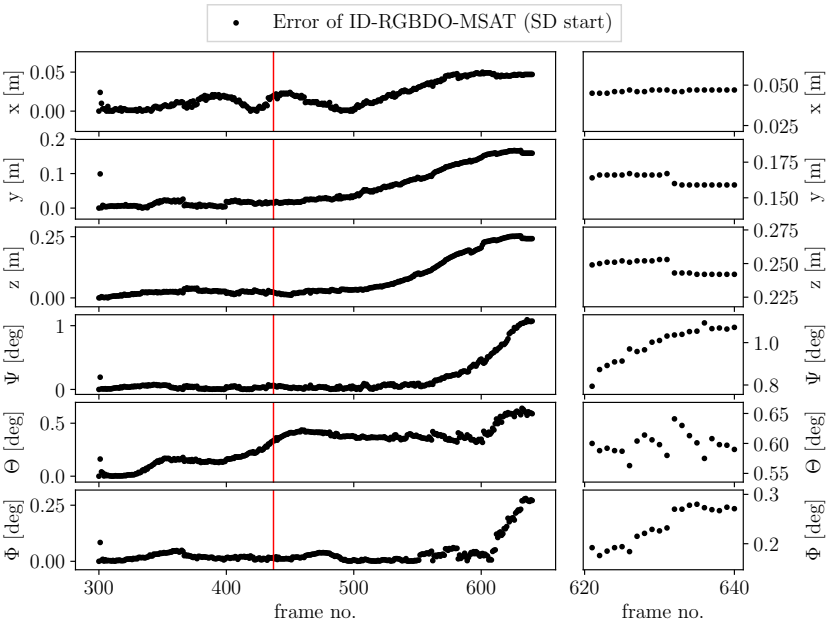


Figure 4.7: The error in rotation and location of the ID-RGBDO-MSAT estimations when running with depth maps estimated by motion stereo. The first detection of an AprilTag is marked with a vertical red line. The odometry was initialised with one simulated depth map. The different scaling of the y-axis must be taken into account when comparing the measured values.

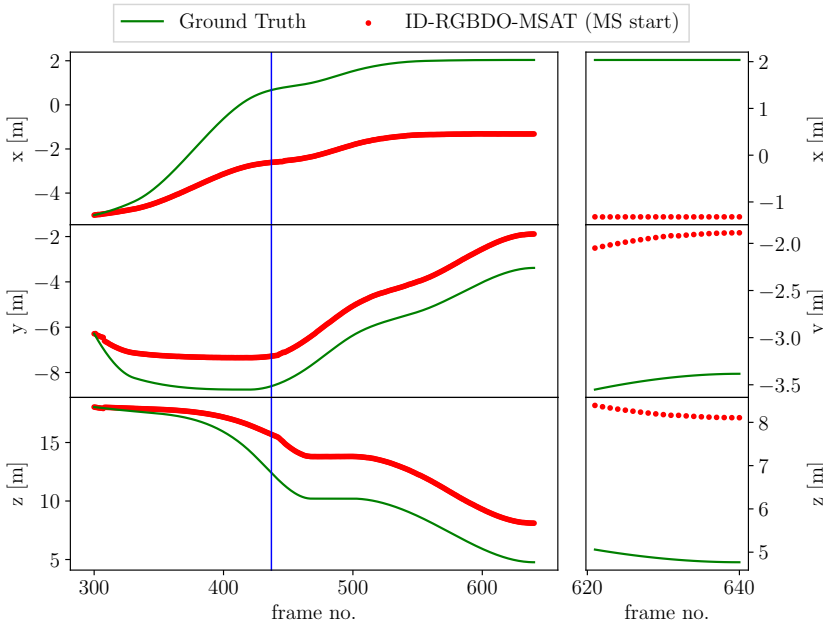


Figure 4.8: The accuracy in location of the ID-RGBDO-MSAT estimations when running with depth maps estimated by motion stereo compared to the ground truth. The odometry was initialised with motion stereo by using a previous image without known transformation.

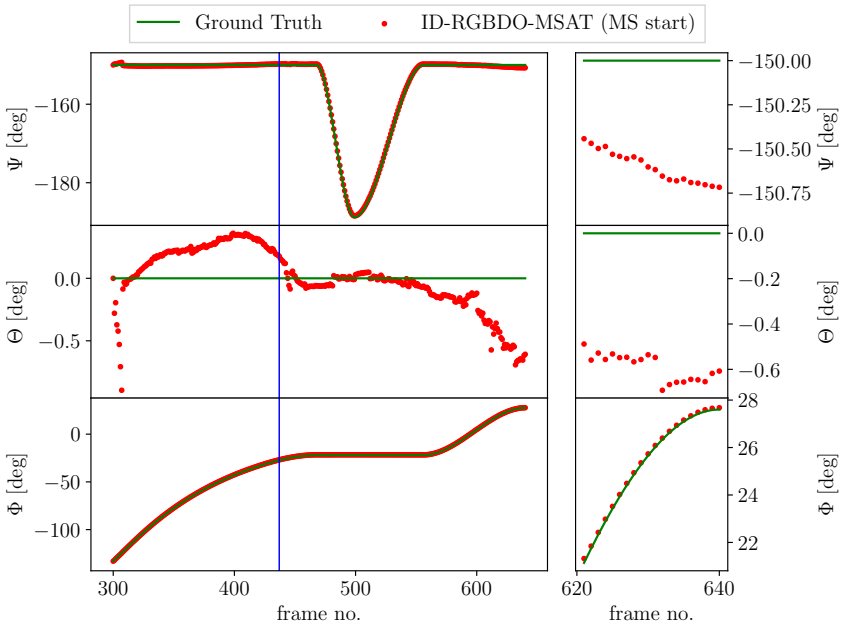


Figure 4.9: The accuracy in rotation of the ID-RGBDO-MSAT estimations when running with depth maps estimated by motion stereo compared to the ground truth. The odometry was initialised with motion stereo by using a previous image without known transformation. The first detection of an AprilTag is marked with a vertical blue line.

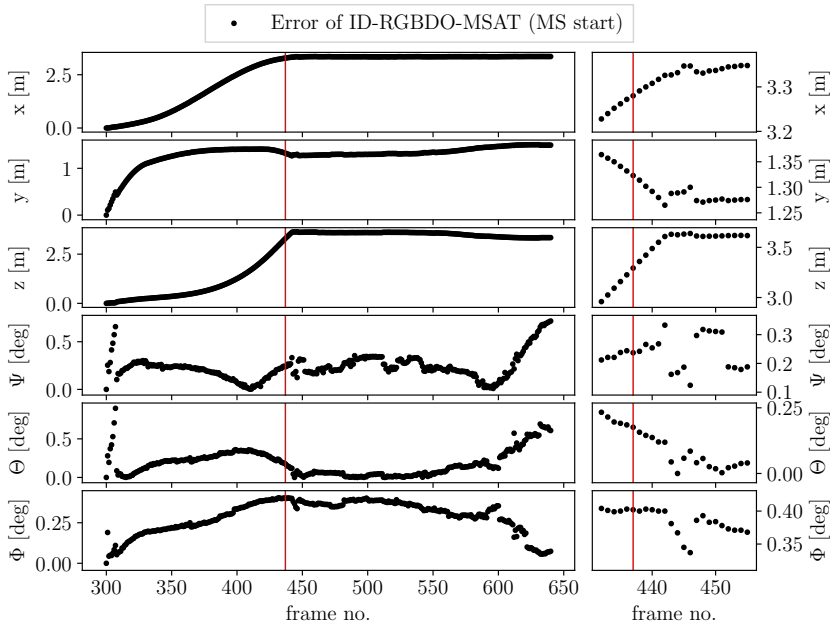


Figure 4.10: The error in rotation and location of the ID-RGBDO-MSAT estimations when running with depth maps estimated by motion stereo. The odometry was initialised with motion stereo by using a previous image without known transformation. The first detection of an AprilTag is marked with a vertical red line.

4.3 Estimating the AprilTag pose with ID-RGBDO-MSAT

The final approach is to express the pose of the AprilTag and its uncertainty in the current frame. It is based on the previously presented approach with the ID-RGBDO-MSAT being started with a motion stereo estimation, and extends it by the combination of the absolute pose estimation of the tag with the relative pose estimation from visual odometry and the respective covariances. The estimation of poses and their uncertainties is shown in figures 4.11 and 4.12. The middle of the three blue lines is the estimation of the AprilTag

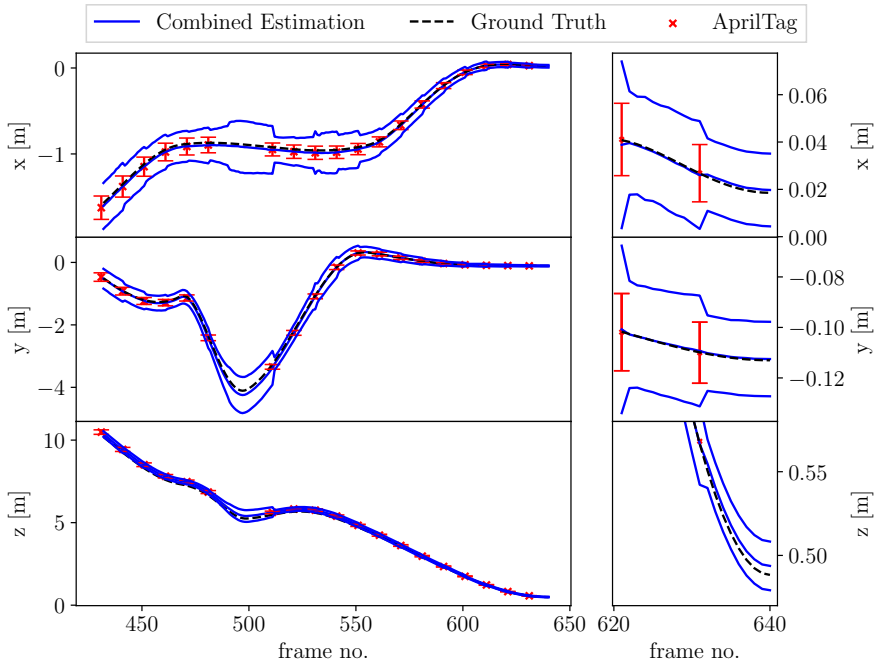


Figure 4.11: The location estimation for the tag with the standard deviation.

pose for all frames, and the upper and lower blue line are the corresponding standard deviation σ_k with $k \in \{x, y, z, \Psi, \Theta, \Phi\}$ of that estimation. The red crosses indicate a measurement of the AprilTag detector and mark the estimated pose. The error bars around the crosses are the standard deviation of the pose estimation from the detection process, which together with the uncertainty of the odometry makes up the outer blue lines. The dotted line is the ground truth for the AprilTag in the camera frame. For the two first tag detections no reliable estimation can be made, as the error is not covered by the model. Therefore this approach works at a distance of approximately $30\times$ the tag-size, which corresponds to $30 \cdot 0.4\,\text{m} = 12\,\text{m}$ in the presented setup, and is slightly below the maximum detection distance of the tag.

In both figures, but more evident in the two plots on the left side in figure 4.11, it is

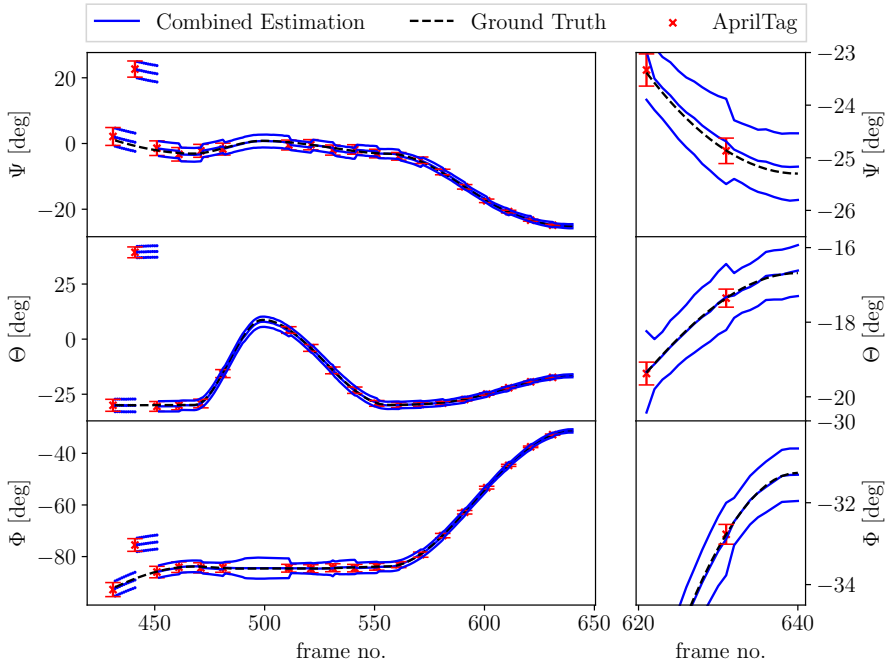


Figure 4.12: The rotation estimation for the tag with the standard deviation.

displayed around frame 500 how the error increases when a tag detection is missing and decreases again when a tag is detected. The standard deviation of the pose estimation in the final frame is $\sigma_{x,y,z} < 2\,\text{cm}$ for the distances and $\sigma_{\Psi,\Theta,\Phi} < 1°$ for the angles.

4.4 Runtime of the ID-RGBDO-MSAT

The program currently still consists of various modules that can only be used together offline. In everyday use, the AprilTag 3 detector alone forms the basis for comparison in terms of running speed, although it has significantly fewer capabilities. In the following the running times for the individual program sections for one frame are shown in section 4.4. The values can be taken as a first starting point for possible improvements. The times were measured several times and averaged.
The estimation of the depthmap from the AprilTag detection increases with the size of the tag in the image. The AprilTag detection on the other hand is faster when the tag is larger. That means this combined steps duration is almost the same all the time. The bottleneck of the algorithm is the estimation of the depthmap with the motion stereo algorithm. In a real world scenario with ARDEA, this step would be computed by a field-programmable gate array (FPGA), which is already on board of the multicopter. The duration for tracking and point selection is done by the ID-RGBDO and was not changed during this thesis. It is only listed for completeness and comparison.
All runtimes were measured by taking only the processing (CPU) time and not the actual wall time. This makes it easier to compare the values when not using multi-core processing. The system it was tested on is a notebook with an Intel Core i5-8265U.

Table 4.1: Average runtime (CPU) per frame of the different modules of ID-RGBDO-MSAT. All values listed are in milliseconds.

	runtime [ms]		
computation step	**max.**	**min.**	**avg.**
AprilTag detection	70	28	55
estimating depthmap (AT)	57	14	25
estimating depthmap (MS)	**723**	**220**	**345**
combining depthmaps	29	23	26
tracking & point selection	**724**	**573**	**604**

Chapter 5

Conclusion

5.1 Discussion and Interpretation

The requirement to develop a method that can be used to track a planar marker and then test it against a data set was met. On the way to this approach, many things were preliminary tried and discarded. The idea to include an Extended Kalman Filter for the direct tracking, was among the first things that came up. The here presented method however is more versatile and profits significantly from the idea of the indirect tracking. As shown in the results chapter, the method yields a good estimation of the markers current pose. What is noticeable form comparing the last approach with the accuracy of the AprilTag is that the error estimation is not optimal, because it still estimates the error higher than the measured one. However, it should be borne in mind that this data was collected using a simulated data set, in which the marker may be better detected than in a real application. Since this approach is to be pursued further, it was therefore necessary to consider the eventuality of a real experiment in advance. The design of the error model was therefore based on this application.
The integration of a stereo matching algorithm that works in all directions of motion and, because it has no fixed baseline, can also be used in different scales has been a great challenge, but has produced very good results.
In summary, this work has touched on many areas and has deepened some of them. In the process, some ideas for possible extensions have been developed, which are briefly outlined below.

5.2 Outlook

Since this approach has so far mainly been a proof-of-concept, it would be interesting to see if it would prove itself in a real-world scenario, running on *Ardea* for example. To accomplish this, first steps have already been taken by preparing some of the modules for

use with the *Robot Operating System (ROS)* (Quigley et al., 2009), which is a framework for structured communication between heterogeneous hardware and software parts, as it is often the case with robots.

5.2.1 Speed up Tag Detection by Tracking

The second approach focused on the acceleration of the detection process of the AprilTag itself. As confirmed by own empirical studies on the AprilTag detector, the segmentation algorithm is the slowest phase in our detection scheme (Olson, 2011). There have been already works concerning this, which tried to change the line detection approach (Romero-Ramirez, Muñoz-Salinas, and Medina-Carnicer, 2018). A typical approach to track an object would be the employment of an Extended Kalman Filter (EKF) or an particle filter (Thrun, 2002). Both filters can be used to track and estimate the 6 degree of freedoms (DoF) of the AprilTag by predicting its position in the next image and then combining its prediction with the actual measurement. The difference between them is that the EKF uses a parametric Gaussian model for the state prediction, whereas the particle filter is a non-parametric filter which uses random state samples for the prediction. An approach for using a particle filter for tracking an AprilTag was done by Wang et al. (2017). With a prediction of the markers pose, the image could be cropped to the estimated area which yields to a faster detection. As Olson states, the correlation between amount of pixels and duration of detection is linear.
The worst case scenario would be that the detector has to run twice; first just on the preselected area and if it fails to find a proper tag, a second time on the whole image. That would increase the computational effort and therefore cause the opposite of the desired goal to speed it up. In order to prevent this from happening as much as possible, the region should thus be chosen generous enough; but on the other hand not too generous, as this would prolong the calculation unnecessarily.
If the AprilTag is close to the camera, the benefit from defining a region of interest will not be as big as if the tag is far away, therefore the parameter to define the search area should be adjusted dynamically and depend on the size of the tag and its distance to the camera.

5.2.2 Changing the model

Instead of the AprilTag it could also be possible to use another object detection module that is based on a 3D model and estimates the 6D position to the camera either via point set registration with an ICP (Pomerleau, Colas, and Siegwart, 2015) or a neural network (Sundermeyer et al., 2018).
There are also neural network based approaches that estimate a dense depth map from a

single image (Eigen, Puhrsch, and Fergus, 2014; Godard et al., 2019). A great advantage of the classical methods in comparison to a neural network is that they do not contain a black box (Nelles, 2001) and uncertainties of the pose can be estimated by propagating measurement errors. A general methodology for estimating reliable uncertainties for the output of deep neural networks has not yet been developed to the knowledge of the author, although there are some recent approaches that show promising results (Loquercio, Segu, and Scaramuzza, 2020; Lee and Triebel, 2020). The challenge here would be to have the necessary hardware on a MAV. In most cases these approaches exceed the capabilities provided by a small aerial vehicle in terms of computational power and energy consumption, as things stand today.

Bibliography

Abbas, Syed Muhammad, Salman Aslam, Karsten Berns, and Abubakr Muhammad. 2019. "Analysis and improvements in apriltag based state estimation." *Sensors (Switzerland)* 19 (24): 1–32. ISSN: 14248220. doi:10.3390/s19245480.

Abidi, M. A., and T. Chandra. 1995. "A New Efficient and Direct Solution for Pose Estimation Using Quadrangular Targets: Algorithm and Evaluation." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17 (5): 534–538. ISSN: 01628828. doi:10.1109/34.391388.

Abraham, Steffen, and Wolfgang Förstner. 2005. "Fish-eye-stereo calibration and epipolar rectification." *ISPRS Journal of photogrammetry and remote sensing* 59 (5): 278–288.

Bergamasco, Filippo, Andrea Albarelli, Emanuele Rodolà, and Andrea Torsello. 2011. "RUNE-Tag: A high accuracy fiducial marker with strong occlusion resilience." *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition:* 113–120. ISSN: 10636919. doi:10.1109/CVPR.2011.5995544.

Blanco Claraco, José Luis. 2010. "A tutorial on SE(3) transformation parameterizations and on-manifold optimization." *University of Malaga, Tech. Rep* 3 (6). http://mapir.isa.uma.es/$%5Csim$jlblanco/papers/jlblanco2010geometry3D_techrep.pdf.

Blender Online Community. 2020. *Blender - a 3D modelling and rendering package.* Blender Institute, Amsterdam: Blender Foundation. http://www.blender.org.

Borenstein, J, L Feng, and B Everett. 1996. "Where am I? Sensors and Methods for Autonomous Mobile Robot Localization."

Bradski, G. 2000. "The OpenCV Library." *Dr. Dobb's Journal of Software Tools.*

Brommer, Christian, Danylo Malyuta, Daniel Hentzen, and Roland Brockers. 2018. "Long-Duration Autonomy for Small Rotorcraft UAS including Recharging." In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 7252–7258. IEEE.

Calonder, Michael, Vincent Lepetit, Christoph Strecha, and Pascal Fua. 2010. "BRIEF: Binary robust independent elementary features." *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6314 LNCS (PART 4): 778–792. ISSN: 16113349. doi:10.1007/978-3-642-15561-1_56.

Chen, Yingfeng, Feng Wu, Wei Shuai, and Xiaoping Chen. 2017. "Robots serve humans in public places—KeJia robot as a shopping assistant." *International Journal of Advanced Robotic Systems* 14 (3): 1–20. ISSN: 17298814. doi:10.1177/1729881417703569.

Cheng, Yang, Mark Maimone, and Larry Matthies. 2005. "Visual odometry on the Mars Exploration Rovers." *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics* 1:903–910. ISSN: 1062922X. doi:10.1109/icsmc.2005.1571261.

Degol, Joseph, Timothy Bretl, and Derek Hoiem. 2017. "ChromaTag: A Colored Marker and Fast Detection Algorithm." *Proceedings of the IEEE International Conference on Computer Vision* 2017-Octob:1481–1490. ISSN: 15505499. doi:10.1109/ICCV.2017.164. arXiv: 1708.02982.

DLR. 2020a. *DLR - Institut für Robotik und Mechatronik - Ardea.* Accessed May 10. https://www.dlr.de/rm/desktopdefault.aspx/tabid-11715/20484_read-47963/.

———. 2020b. *Micro Aerial Vehicle (MAV) Ardea (2016).* Accessed May 10. https://www.dlr.de/rm/Portaldata/52/Resources/roboter_und_systeme/ardea/ardea_2016_1440x810.jpg.

———. *Heterogenes Roboter-Team Ardea und LRU.* https://www.dlr.de/rm/Portaldata/52/Resources/roboter_und_systeme/ardea/ardea_LRU_2016_1440x810.jpg.

Eigen, David, Christian Puhrsch, and Rob Fergus. 2014. "Depth map prediction from a single image using a multi-scale deep network." *Advances in Neural Information Processing Systems* 3 (January): 2366–2374. ISSN: 10495258.

Engel, Jakob, Vladlen Koltun, and Daniel Cremers. 2018a. "Direct Sparse Odometry." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40 (3): 611–625. ISSN: 01628828. doi:10.1109/TPAMI.2017.2658577. arXiv: 1607.02565.

———. 2018b. "Direct Sparse Odometry." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40 (3): 611–625. ISSN: 01628828. doi:10.1109/TPAMI.2017.2658577. arXiv: 1607.02565.

Everett, H R. 1995. *Sensors for mobile robots.* CRC Press.

Fiala, Mark. 2005. "ARTag, a fiducial marker system using digital techniques." *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005* II:590–596. doi:10.1109/CVPR.2005.74.

Fontan, Alejandro, Javier Civera, and Rudolph Triebel. 2020. "Information-Driven Direct RGB-D Odometry." In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

Förstner, Wolfgang, and Eberhard Gülch. 1987. "A fast operator for detection and precise location of distinct points, corners and centres of circular features." In *Proc. ISPRS intercommission conference on fast processing of photogrammetric data,* 281–305. Interlaken.

Fraundorfer, Friedrich, and Davide Scaramuzza. 2012. "Visual odometry: Part II: Matching, robustness, optimization, and applications." *IEEE Robotics and Automation Magazine* 19 (2): 78–90. ISSN: 10709932. doi:10.1109/MRA.2012.2182810.

Gillies, David. 2015. *Close Range Photogrammetry,* 30:318–322. 151. ISBN: 9781849950572. doi:10.1111/phor.12114.

Godard, Clement, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. 2019. "Digging into self-supervised monocular depth estimation." *Proceedings of the IEEE International Conference on Computer Vision* 2019-Octob (1): 3827–3837. ISSN: 15505499. doi:10.1109/ICCV.2019.00393.

Harris, Christopher G, Mike Stephens, et al. 1988. "A combined corner and edge detector." In *Alvey vision conference,* 15:10–5244. 50. Citeseer.

Hartley, Richard, and Andrew Zisserman. 2003. *Multiple View Geometry in Computer Vision.* Cambridge university press.

Howard, Andrew. 2008. "Real-time stereo visual odometry for autonomous ground vehicles." In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 3946–3952. IEEE.

Jin, Pengju, Pyry Matikainen, and Siddhartha S. Srinivasa. 2017. "Sensor fusion for fiducial tags: Highly robust pose estimation from single frame RGBD." *IEEE International Conference on Intelligent Robots and Systems* 2017-Septe:5770–5776. ISSN: 21530866. doi:10.1109/IROS.2017.8206468.

JPL. *Curiosity (Dirty).* https://nasa3d.arc.nasa.gov/detail/curiosity-dirty.

Karami, Ebrahim, Siva Prasad, and Mohamed Shehata. 2017. "Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images." *arXiv preprint arXiv:1710.02726.*

Kato, H., and M. Billinghurst. 1999. "Marker tracking and HMD calibration for a video-based augmented reality conferencing system." *Proceedings - 2nd IEEE and ACM International Workshop on Augmented Reality, IWAR 1999:* 85–94. doi:10.1109/IWAR.1999.803809.

Kerl, C, J Sturm, and D Cremers. 2013. "Robust Odometry Estimation for RGB-D Cameras." In *International Conference on Robotics and Automation (ICRA).*

Kraus, Karl. 2007. *Photogrammetry - Geometry from Images and Laser Scans,* 53:1689–1699. 9. ISBN: 9788578110796. doi:10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

Krogius, Maximilian, Acshi Haggenmiller, and Edwin Olson. 2019. "Flexible Layouts for Fiducial Tags." *IEEE International Conference on Intelligent Robots and Systems:* 1898–1903. ISSN: 21530866. doi:10.1109/IROS40897.2019.8967787.

Kunz, Christian, Vera Genten, Pascal Meissner, and Björn Hein. 2019. "Metric-based evaluation of fiducial markers for medical procedures." In *Medical Imaging 2019: Image-Guided Procedures, Robotic Interventions, and Modeling,* 10951:97. International Society for Optics and Photonics. ISBN: 9781510625495. doi:10.1117/12.2511720.

Lee, Jongseok, Ribin Balachandran, Yuri S. Sarkisov, Marco De Stefano, Andre Coelho, Kashmira Shinde, Min Jun Kim, Rudolph Triebel, and Konstantin Kondak. 2020. "Visual-Inertial Telepresence for Aerial Manipulation." arXiv: 2003.11509. http://arxiv.org/abs/2003.11509.

Lee, Jongseok, and Rudolph Triebel. 2020. *Representing Model Uncertainty of Neural Networks in Sparse Information Form.* https://openreview.net/forum?id=Bkxd9JBYPH.

Liang, Pengpeng, Yifan Wu, Hu Lu, Liming Wang, Chunyuan Liao, and Haibin Ling. 2018. "Planar object tracking in the wild: A benchmark." *Proceedings - IEEE International Conference on Robotics and Automation:* 651–658. ISSN: 10504729. doi:10.1109/ICRA.2018.8461037. arXiv: 1703.07938.

López-Cerón, Alberto, and José M. Cañas. 2016. "Accuracy analysis of marker-based 3D visual localization." *XXXVII Jornadas de Automatica Workshop.* http://people.csail.mit.edu/kaess/apriltags.

Loquercio, Antonio, Mattia Segu, and Davide Scaramuzza. 2020. "A General Framework for Uncertainty Estimation in Deep Learning." *IEEE Robotics and Automation Letters* 5 (2): 3153–3160. ISSN: 23773766. doi:10.1109/LRA.2020.2974682. arXiv: 1907.06890.

Lorenz, Ralph D, Elizabeth P Turtle, Jason W Barnes, Melissa G Trainer, Douglas S Adams, Kenneth E Hibbard, Colin Z Sheldon, Kris Zacny, Patrick N Peplowski, David J Lawrence, et al. 2018. "Dragonfly: a Rotorcraft Lander Concept for scientific exploration at Titan." *Johns Hopkins APL Technical Digest.*

Lowe, David G. 2004. "Distinctive image features from scale-invariant keypoints." *International journal of computer vision* 60 (2): 91–110.

Lucas, Bruce D, Takeo Kanade, et al. 1981. "An iterative image registration technique with an application to stereo vision."

Lutz, Philipp, Marcus G. Müller, Moritz Maier, Samantha Stoneman, Teodor Tomić, Ingo von Bargen, Martin J. Schuster, et al. 2020. "ARDEA—An MAV with skills for future planetary missions." *Journal of Field Robotics,* no. May 2019: 515–551. ISSN: 15564967. doi:10.1002/rob.21949.

Maybank, Stephen. 1993. *Theory of Reconstruction from Image Motion.* 261. ISBN: 9783642775598.

Möbius, August Ferdinand. 1827. *Der barycentrische Calcul: ein neues Hülfsmittel zur analytischen Behandlung der Geometrie.* Barth.

Moravec, Hans P. 1980. "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover, Tech. Report, Robotics Institute, Carnegie-Mellon University, pp.1-175": 1–175. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA092604.

———. 1981. "Rover Visual Obstacle Avoidance." 2:785–790.

Müller, M. G., F. Steidle, M. J. Schuster, P. Lutz, M. Maier, S. Stoneman, T. Tomic, and W. Sturzl. 2018. "Robust Visual-Inertial State Estimation with Multiple Odometries and Efficient Mapping on an MAV with Ultra-Wide FOV Stereo Vision." *IEEE International Conference on Intelligent Robots and Systems:* 3701–3708. ISSN: 21530866. doi:10.1109/IROS.2018.8594117.

Munoz-Salinas, Rafael, Manuel J Marin-Jimenez, and Rafael Medina-Carnicer. 2019. "SPM-SLAM: Simultaneous localization and mapping with squared planar markers." *Pattern Recognition* 86:156–171.

Mur-Artal, Raul, J. M.M. Montiel, and Juan D. Tardos. 2015. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System." *IEEE Transactions on Robotics* 31 (5): 1147–1163. ISSN: 15523098. doi:10.1109/TRO.2015.2463671. arXiv: 1502.00956.

Negahdaripour, Shahriar, Ricard Prados, and Rafael Garcia. 2005. "Planar homography: Accuracy analysis and applications." *Proceedings - International Conference on Image Processing, ICIP* 1 (May 2014): 1089–1092. ISSN: 15224880. doi:10.1109/ICIP.2005.1529944.

Nelles, Oliver. 2001. *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models.* Springer Science & Business Media.

Nissler, Christian, Stefan Buttner, Zoltan Csaba Marton, Laura Beckmann, and Ulrike Thomasy. 2016. "Evaluation and improvement of global pose estimation with multiple AprilTags for industrial manipulators." *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* 2016-Novem. ISSN: 19460759. doi:10.1109/ETFA.2016.7733711.

Nistér, David, Oleg Naroditsky, and James Bergen. 2004. "Visual odometry." *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 1. ISSN: 10636919. doi:10.1109/cvpr.2004.1315094.

Olson, Edwin. 2011. "AprilTag: A robust and flexible visual fiducial system." *Proceedings - IEEE International Conference on Robotics and Automation:* 3400–3407. ISSN: 10504729. doi:10.1109/ICRA.2011.5979561.

Pathak, Sarthak, Alessandro Moro, Atsushi Yamashita, and Hajime Asama. 2016. "Dense 3D reconstruction from two spherical images via optical flow-based equirectangular epipolar rectification." *IST 2016 - 2016 IEEE International Conference on Imaging Systems and Techniques, Proceedings:* 140–145. doi:10.1109/IST.2016.7738212.

Pearson, Karl. 1901. "On lines and planes of closest fit to systems of points in space." *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2 (11): 559–572. ISSN: 1941-5982. doi:10.1080/14786440109462720.

Pfrommer, Bernd, and Kostas Daniilidis. 2019. "TagSLAM: Robust SLAM with fiducial markers." *arXiv preprint arXiv:1910.00679.*

Pfrommer, Bernd, Nitin Sanket, Kostas Daniilidis, and Jonas Cleveland. 2017. "Penncosyvio: A challenging visual inertial odometry benchmark." In *2017 IEEE International Conference on Robotics and Automation (ICRA),* 3847–3854. IEEE.

Pollefeys, Marc, Reinhard Koch, and Luc Van Gool. 1999. "Simple and efficient rectification method for general motion." *Proceedings of the IEEE International Conference on Computer Vision* 1 (August 2014): 496–501. doi:10.1109/iccv.1999.791262.

Pomerleau, François, Francis Colas, and Roland Siegwart. 2015. "A Review of Point Cloud Registration Algorithms for Mobile Robotics." *A Review of Point Cloud Registration Algorithms for Mobile Robotics* X (X). doi:10.1561/9781680830255.

Quigley, Morgan, Ken Conley, Brian P Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. 2009. "ROS: an open-source Robot Operating System." In *ICRA Workshop on Open Source Software.*

Romero-Ramirez, Francisco J., Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. 2018. "Speeded up detection of squared fiducial markers." *Image and Vision Computing* 76 (June): 38–47. ISSN: 02628856. doi:10.1016/j.imavis.2018.05.004.

Rosin, Paul L. 1999. "Measuring Corner Properties." *Computer Vision and Image Understanding* 73 (2): 291–307. ISSN: 10773142. doi:10.1006/cviu.1998.0719.

Rosten, Edward, and Tom Drummond. 2006. "Machine learning for high-speed corner detection." In *European conference on computer vision,* 430–443. Springer.

Rublee, Ethan, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. "ORB: An efficient alternative to SIFT or SURF." *Proceedings of the IEEE International Conference on Computer Vision:* 2564–2571. doi:10.1109/ICCV.2011.6126544.

Sagitov, Artur, Ksenia Shabalina, Leysan Sabirova, Hongbing Li, and Evgeni Magid. 2017. "ARTag, AprilTag and CALTag fiducial marker systems: Comparison in a presence of partial marker occlusion and rotation." *ICINCO 2017 - Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics* 2 (January 2018): 182–191. doi:10.5220/0006478901820191.

Scaramuzza, Davide, and Friedrich Fraundorfer. 2011. "Tutorial: Visual odometry." *IEEE Robotics and Automation Magazine* 18 (4): 80–92. ISSN: 10709932. doi:10.1109/MRA.2011.943233.

Schmidt Jr., Rodney Albert. 1971. *A Study of the Real-Time Control of a Computer Driven Vehicle.* Technical report. STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE.

Schuler, Tristan, Greta Studier, and Tom Bryan. 2018. "Developed AprilNav, an Indoor Navigation and Localization System for Autonomous Testing of Electric Sail Dynamics."

Schuster, Martin J, G M Marcus, Sebastian G Brunner, Hannah Lehner, Peter Lehner, D Andreas, Mallikarjuna Vayugundla, et al. 2019. "Towards Heterogeneous Robotic Teams for Collaborative Scientific Sampling in Lunar and Planetary Environments." *Workshop on Informed Scientific Sampling in Large-scale Outdoor Environments at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* no. February 2020.

Schuster, Martin Johannes. 2019. "Collaborative Localization and Mapping for Autonomous Planetary Exploration: Distributed Stereo Vision-Based 6D SLAM in GNSS-Denied Environments." PhD diss., Universität Bremen.

Schweighofer, Gerald, and Axel Pinz. 2006. "Robust pose estimation from a planar target." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (12): 2024–2030. ISSN: 01628828. doi:10.1109/TPAMI.2006.252.

Shi, Jianbo, et al. 1994. "Good features to track." In *1994 Proceedings of IEEE conference on computer vision and pattern recognition,* 593–600. IEEE.

Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. 2011. *Introduction to autonomous mobile robots.* MIT press.

Srinivasan, M, Shaowu Zhang, M Lehrer, and T S Collett. 1996. "Honeybee navigation en route to the goal: visual flight control and odometry." *Journal of Experimental Biology* 199 (1): 237–244.

Sundermeyer, Martin, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. 2018. "Implicit 3d orientation learning for 6d object detection from rgb images." In *Proceedings of the European Conference on Computer Vision (ECCV),* 699–715.

Tareen, Shaharyar Ahmed Khan, and Zahra Saleem. 2018. "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK." In *2018 International conference on computing, mathematics and engineering technologies (iCoMET),* 1–10. IEEE.

Thrun, Sebastian. 2002. "Probabilistic robotics." *Communications of the ACM* 45 (3): 52–57. ISSN: 00010782. doi:10.1145/504729.504754.

Tomasi, Carlo, and Takeo Kanade. 1991. "Detection and tracking of point features."

Vetter, Sebastian. 2015. "Fehleranalyse und -modellierung markerbasierter 6D-Posenschätzungen."

Wagner, Daniel, and Dieter Schmalstieg. 2007. "Artoolkitplus for pose tracking on mobile devices."

Wang, John, and Edwin Olson. 2016. "AprilTag 2: Efficient and robust fiducial detection." *IEEE International Conference on Intelligent Robots and Systems* 2016-Novem:4193–4198. ISSN: 21530866. doi:10.1109/IROS.2016.7759617.

Wang, Ju, Chad Sadler, Cesar Flores Montoya, and Jonathan C.L. Liu. 2017. "Optimizing ground vehicle tracking using unmanned aerial vehicle and embedded Apriltag design." *Proceedings - 2016 International Conference on Computational Science and Computational Intelligence, CSCI 2016:* 739–744. doi:10.1109/CSCI.2016.0144.

Zabih, Ramin, and John Woodfill. 1994. "Non-parametric local transforms for computing visual correspondence." In *European conference on computer vision,* 151–158. Springer.

Appendix

I Measurements

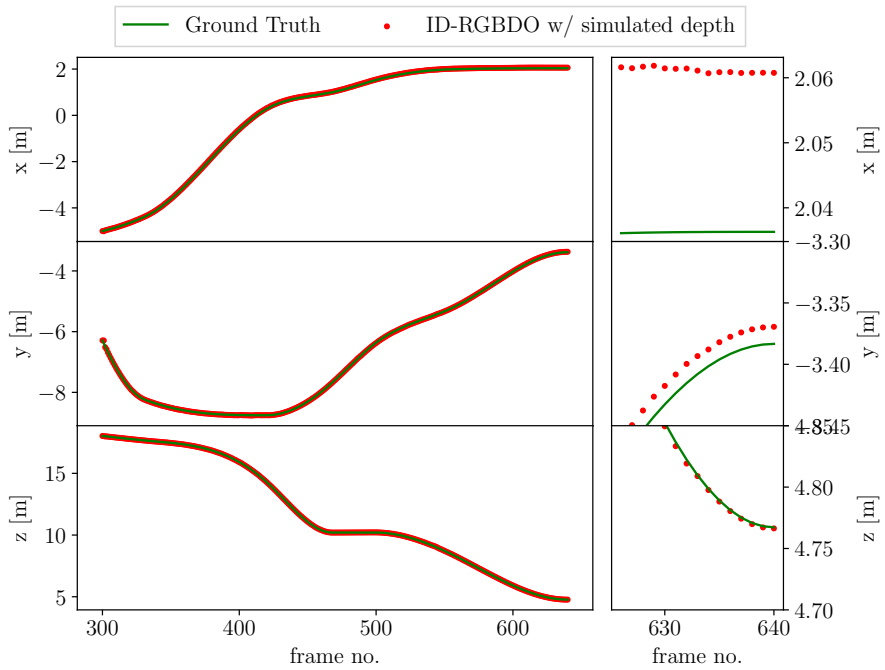


Figure 5.1: The accuracy in location of the ID-RGBDO measurements when running with simulated depth maps compared to the ground truth. The different scaling of the y-axis must be taken into account when comparing the measured values.

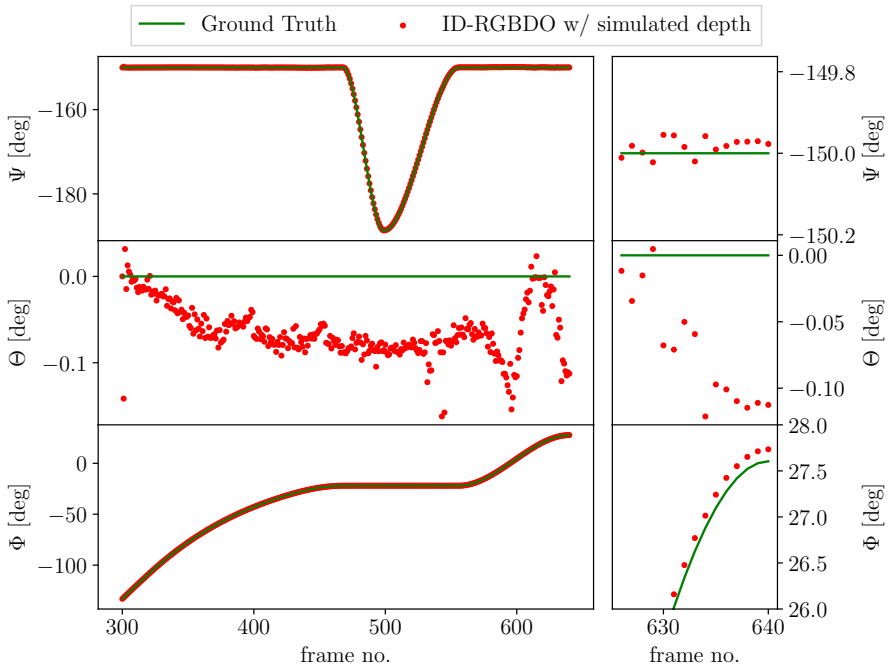


Figure 5.2: The accuracy in rotation of the ID-RGBDO measurements when running with simulated depth maps compared to the ground truth. The different scaling of the y-axis must be taken into account when comparing the measured values.